Universidad Nacional Autónoma de México
Semestre 2020-2
Compiladores
Juárez Aguilar Osmar
Méndez Cabrera Ana Belem
Morales Garcia Luis Angel
Rodríguez Sánchez José Andrés

Definición Dirigida por Sintaxis

(a) Gramática

----*sin: significa sin tipo, car: tipo caracter*-----

1. programa → declaraciones funciones

2. declaraciones → tipo lista_var**;** declaraciones | tipo_registro lista_var**;** declaraciones | ε

3. tipo_registro → **estructura inicio** declaraciones **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 → **def** tipo **id(**argumentos**) inicio** declaraciones sentencias **fin** funciones | ε

9. argumentos → lista_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 sentencia | sentencia

15. sentencia → **si** e_bool **entonces** sentencia **fin**
| **si** e_bool **entonces** sentencia **sino** sentencia **fin**
| **mientras** e_bool **hacer** sentencia **fin**
| **hacer** sentencia **mientras** e_bool**;**
| **segun (**variable**) hacer** casos predeterminado **fin**
| variable **:=** expresion **;**
| **escribir** expresion **;**
| **leer** variable **;**
| **devolver;**
| **devolver** expresion**;**
| **terminar;**
| **inicio** sentencias **fin**

16. casos → **caso num:** sentencia casos | **caso num:** sentencia

17. predeterminado → **pred:** sentencia | ε

18. e_bool → e_bool **o** e_bool | e_bool **y** e_bool | **no** e_bool | **(** e_bool **)** | relacional
| **verdadero** | **falso**

19. relacional → relacional > relacional
| relacional < relacional
| relacional <= relacional
| relacional >= relacional
| relacional <> relacional
| relacional = relacional
| expresion

20. expresion → expresion + expresion
| expresion − expresion
| expresion ∗ expresion
| expresion / expresion
| expresion **%** expresion | **(**expresion**)**
| variable | **num** | **cadena** | **carácter**

21. expresion → **id** variable_comp

22. variable_comp → dato_est_sim | arreglo | **(** parametros **)**

23. dato_est_sim → dato_est_sim **.id** | ε

24. arreglo → **[** expresion **]** | arreglo **[** expresion **]**

25. parametros → lista_param | ε

26. lista_param → lista_param**,** expresion | expresion

(b) Definición dirigida por sintaxis

programa→ declaraciones funciones	dir = 0 SSTACK = init_sym_tab_stack() TSTACK = init_type_tab_stack() SYMTAB = init_sym_tab() TYPTAB = init_typ_tab() SSTACK.push_st(SYMTAB) TSTACK.push_st(TYMTAB) TabCad = newTablaCad()
declaraciones→ tipo lista_var; declaraciones	TYP = tipo.tipo
declaraciones→ tipo_registro lista_var**;** declaraciones	TYP = tipo _registro.tipo
tipo_registro→ **estructura inicio** declaraciones **fin**	SYMTAB = init_sym_tab() inicio declaraciones fin TYPSYM = init_typ_tab() Sdir.push(dir)********** dir = 0 TSTACK.push_st(TYPTAB)

	SSTACK.push_st(SYMTAB) dir = Sdir.pop()********** TYPTAB1 = TSTACK.pop_st() SSTACK.getTop().setType(TYPTAB1) SYMTAB1 = SSTACK.pop_st() dir = Sdir.pop()********** TYP=TSTACK.getTop().append_type (“estructura”,0,TYPTAB1)
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	Si num.tipo = ent y num.val > 0 entonces tipo_arreglo.tipo = TSTACK.getTop().append_type(“array”, num.val,tipo_arreglo.tipo) sino Error(“El indice tiene que ser entero y mayor a 0”)
tipo_arreglo→ ε	tipo_arreglo.tipo = base
lista_var→ lista_var**, id**	si SSTACK.getTop().getId(id.lexval) = -1 entonces SSTACK.getTop().append_sym(id.lexval,tipo, dir, ”var”) dir = dir + TSTACK.getTop().getTam(tipo) sino Error(”El identificador ya fue declarado”)
lista_var→ **id**	si SSTACK.getTop().getId(id.lexval) = -1 entonces SSTACK.getTop().append_sym(id.lexval,tipo,dir,”var”) dir=dir+TSTACK.getTop().getTam(tipo) sino Error(”El identificador ya fue declarado”)
funciones→**def** tipo **id(**argumentos**) inicio** declaraciones sentencias **fin** funciones	si SSTACK.getTail().getId(id.lexval) = -1 entonces SSTACK.getTop().append_sym(id.lexval, tipo, −−, ”def”) Sdir.push(dir) FuncType = tipo.tipo FuncReturn = false dir = 0 TSTACK.push_st (TYPTAB) SSTACK.push_st (SYMTAB) dir = Sdir.pop() add quad(code,0label0, −, −, id.lexval) L = newLabel()

	backpatch(code, sentencias.next, L) add quad(code,0label0, −, −, L) TSTACK.pop_st () SSTACK.pop_st () dir = Sdir.pop() SSTACK.getTop().append_arg(id.lexval, argumentos.lista) si (tipo.tipo = sin) y (FuncReturn = false) entonces Error(la funcíon no tiene valor de retorno) fin si sino Error(”El identificador ya fue declarado”)
argumentos→ lista_arg	argumentos.lista = lista.arg.lista
argumentos → **sin**	argumentos.lista = nulo
lista_arg→ lista_arg**,** arg	lista arg.lista = lista arg1.lista lista arg.lista.add(arg.tipo) lista arg.num = lista arg1.num +1
lista_arg→ arg	lista arg.lista = newListaParam()********** lista arg.lista.append(arg.tipo)
arg→ tipo_arg **id**	si SSTACK.getTop().getId(id.lexval) = -1 entonces SSTACK.getTop().append_sym(id.lexval, tipo, dir, ”var”) dir = dir + TSTACK.getTop().getTam(tipo) sino Error(”El identificador ya fue declarado”)
tipo_arg→ base param_arr	base = base.tipo tipo_arg = param_arr.tipo
param_arr→ **[ ]** param_arr	param arr.tipo = TSTACK.getTop().append_type(”array”, -, param arr.tipo)
param_arr→ ε	param_arr.tipo = base
sentencias→ sentencias sentencia	L = newLabel() backpatch(code, sentecias.listnext, L) sentencias.listnext = sentencia.listnext
sentencias→ sentencia	sentencias.listnext = sentencia.listnext
sentencia→ **si** e_bool **entonces** sentencia **fin**	L = newLabel() backpatch(code, e_bool.listtrue, L) sentencia.listnext=combinar(e_bool.listfalse,sentencia.listnext)
sentencia→ **si** e_bool **entonces** sentencia **sino** sentencia **fin**	
sentencia→ **mientras** e_bool **hacer** sentencia **fin** *******	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(sentencia1.listnext, L1) backpatch(e-bool.listtrue, L2) sentencia.nextlist = e_bool.listfalse sentencia.code = etiqueta(L1)

	‖e_bool.code ‖ etiqueta(L2) ‖ sentencia.code ‖ gen(‘goto’ sentencia.listnext[0]) ‖ sentencia1.code
sentencia→ **hacer** sentencia **mientras** e_bool**;**	L = newLabel() backpatch(code, e_bool.listtrue, L) backpatch(code, sentencia.listnext, L1) sentencia.listnext = e_bool.listfalse add quad(code, ”label”, -, -, L)
sentencia→ **segun (**variable**) hacer** casos predeterminado **fin**	
sentencia→ variable **:=** expresion **;**	α = reducir(expresion.dir, expresion.tipo, variable.tipo) add quad(code, ” = ”, α, −, 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”, −, −, −) sino Error(”La funcion debe retornar algun 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 else Error(”La funcion no puede retornar algun valor de tipo ”) fin sentencia.listnext= nulo
sentencia→ **terminar;**	I = newIndex() add quad(code, ”goto”, -, -, I) sentencia.listnext = newList() sentencia.listnext.add(I)
sentencia→ **inicio** sentencias **fin**	
casos→ **caso num:** sentencia casos	backpatch(caso) casos.dir=sentencia.dir casos.val=setencia.val
casos→ **caso num:** sentencia	backpatch(casos) casos.dir=casos1.dir+setencias.dir

predeterminado→ **pred:** sentencia	predeterminado.dir=sentencias.dir
predeterminado→ ε	predeterminado.dir=null
e_bool→ e_bool **o** e_bool	L = newLabel() backpatch(code, e_bool1.listfalse, L) e_bool.listtrue=combinar(e_bool1.listtrue,e_bool2.lisstrue) e_bool.listfalse= e_bool2.listfalse add quad(code, ”label”, -, -, L)
e_bool→ e_bool **y** e_bool	L = newLabel() backpatch(code, e_bool1.listtrue, L) e_bool.listtrue =e_bool2.listtrue e_bool.listfalse=combinar(e_bool1.listfalse,e_bool2.listfalse) add quad(code, ”label”, -, -, L)
e_bool→ **no** e_bool	e_bool.listtrue = e_bool1.listfalse e_bool.listfalse = e_bool1.listtrue
e_bool→ **(** e_bool **)**	e_bool.listtrue = e_bool1.listtrue e_bool.listfalse = e_bool1.listfalse
e_bool→ relacional	e_bool.listtrue=relacional.listtrue e_bool.listfalse = relacional.listfalse
e_bool→ **verdadero**	I=newIndex() e_bool.listtrue=newList() e_bool.listtrue.add(I) add quad(code, ”goto”, -, -, I) e_bool.listfalse = nulo
e_bool→ **falso**	I = newIndex() e_bool.listtrue = nulo e_bool.listfalse = newList() e_bool.listfalse.add(I) add quad(code, ”goto”, -, -, I)
relacional→ relacional > relacional	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo=max(relacional1.tipo, relacional2.tipo) α1=ampliar(relacional1.dir,relacional1.tipo,relacional.tipo) α2=ampliar(relacional2.dir,relacional2.tipo,relacional.tipo) add quad(code, ”>”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1)
relacional→relacional < relacional	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I)

	relacional.listfalse.add(I1) relacional.tipo= max(relacional1.tipo, relacional2.tipo) α1=ampliar(relacional1.dir,relacional1.tipo,relacional.tipo) α2=ampliar(relacional2.dir,relacional2.tipo,relacional.tipo) add quad(code, ”<”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1)
relacional→ relacional <= relacional	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo=max(relacional1.tipo, relacional2.tipo) α1=ampliar(relacional1.dir,relacional1.tipo,relacional.tipo) α2=ampliar(relacional2.dir,relacional2.tipo,relacional.tipo) add quad(code, ”<=”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1)
relacional→ relacional >= relacional	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo=max(relacional1.tipo, relacional2.tipo) α1 = ampliar(relacional1.dir, relacional1.tipo, relacional.tipo) α2 = ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) add quad(code, ”>=”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1)
relacional→ relacional <> relacional	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo=max(relacional1.tipo, relacional2.tipo) α1=ampliar(relacional1.dir,relacional1.tipo,relacional.tipo) α2=ampliar(relacional2.dir,relacional2.tipo,relacional.tipo) add quad(code, ”<>”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1)
relacional→ relacional = relacional	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I)

	relacional.listfalse.add(I1) relacional.tipo=max(relacional1.tipo, relacional2.tipo) α1=ampliar(relacional1.dir,relacional1.tipo,relacional.tipo) α2=ampliar(relacional2.dir,relacional2.tipo,relacional.tipo) add quad(code, ”=”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1)
relacional→ expresion	relacional.tipo=expresion.tipo relacional.dir = expresion.dir
expresion→ expresion + expresion	expresion.tipo = max(expresion1.tipo,expresion2.tipo) expresion.dir = newTemp() α1=ampliar(expresion1.dir,expresion1.tipo,expresion.tipo) α2=ampliar(expresion2.dir,expresion2.tipo,expresion.tipo) add quad(code, ”+”,α1 ,α2, expresion.dir)
expresion→ expresion − expresion	expresion.tipo = max(expresion1.tipo,expresion2.tipo) expresion.dir = newTemp() α1=ampliar(expresion1.dir,expresion1.tipo,expresion.tipo) α2=ampliar(expresion2.dir,expresion2.tipo,expresion.tipo) add quad(code, ”–”,α1 ,α2, expresion.dir)
expresion→ expresion ∗ expresion	expresion.tipo = max(expresion1.tipo,expresion2.tipo) expresion.dir = newTemp() α1=ampliar(expresion1.dir,expresion1.tipo,expresion.tipo) α2=ampliar(expresion2.dir,expresion2.tipo,expresion.tipo) add quad(code, ”∗”,α1 ,α2, expresion.dir)
expresion→ expresion / expresion	expresion.tipo = max(expresion1.tipo,expresion2.tipo) expresion.dir = newTemp() α1=ampliar(expresion1.dir,expresion1.tipo,expresion.tipo) α2=ampliar(expresion2.dir,expresion2.tipo,expresion.tipo) add quad(code, ”/”,α1 ,α2, expresion.dir)
expresion→ expresion **%** expresion	expresion.tipo = max(expresion1.tipo,expresion2.tipo) expresion.dir = newTemp() α1=ampliar(expresion1.dir,expresion1.tipo,expresion.tipo) α2=ampliar(expresion2.dir,expresion2.tipo,expresion.tipo) add quad(code, ”%”,α1 ,α2, expresion.dir)
expresion→ **(**expresion**)**	expresion.dir = expresion1.dir expresion.tipo = expresion1.tipo
expresion→ variable	epxreson.dir = newTemp() expresion.tipo = variable.tipo

	add quad(code, ”∗”, variable.base[variable.dir] , -, expresion.dir)
expresion→ **num**	expresion.tipo = num.tipo expresion.dir = num.val
expresion→ **cadena**	expresion.tipo = cadena expresion.dir = TabCad.add(cadena)
expresion→ **caracter**	expresion.tipo = caracter expresion.dir = TabCad.add(caracter)
variable→ **id** variable_comp	
variable_comp→ dato_est_sim	
variable_comp→ arreglo	
variable_comp→ **(** parametros **)**	
dato_est_sim→ dato_est_sim**.id**	
dato_est_sim→ ε	
arreglo→ **[** expresion **]**	
arreglo→ arreglo **[** expresion **]**	
parametros→ lista_param	parametros.tipos=lista_param.tipos
parametros→ ε	parametros.tipos=nulo
lista_param→ lista_param**,** expresion	lista_param.tipos = lista_param
lista_param→ expresion	lista_param.tipos = expression.tipos lista_param.dir = expression.dir