Definición dirigida por Sintaxis

Álvarez González Ian

García Oviedo Jaasiel Osmar

López López Ulysses

Domínguez Cisneros Alexis Saul

Gpo 1.

Compiladores |

2020-2

Glosario

Abreviación	Significado
P	Programa
D_F	Declaraciones funciones
D	Declaraciones
TL_V	Tipo lista_var
T_R	Tipo_registro
T	Tipo
T_A	Tipo_arreglo
B	Base
Ent	Entero
Real	Real
Dreal	Double
Car	Tipo carácter
Sin	Sin tipo
T_A	Tipo_arreglo
Num	Numero
L_V	Lista_var
Id	Id
F	Funciones
ARG	Argumentos
S	Sentencias
L_ARG	Lista_arg
T_ARG	Tipo_arg
PA	Param_arr
P_A	Param_arr
E_B	E_bool
V	Variable
CP	Casos predeterminados
E	Expresión
CAS	Caso
CASS	Casos
PRED	Predeterminado
REL	Relacional
||	O
&&	Y
!	No
V_C	Variable_comp

DST	Dato_est_sim
A	Arreglo
P	Parametros
L_P	Lista_par
STS	Stack_table_symbols
STT	Stack_table_type
TS	Table_symbols
TT	Table_type
TOS	Table_of_strings
L	Lista
L_R	Lista_retorno
S_D	Stack_dir

Definición dirigida por sintaxis

Regla de Producción	Regla Semántica
1) P → D_F	STS.push(nuevaTS()) STT.push(nuevaTT()) dir = 0 P.codigo = S.codigo TOS = nuevaTOS()
2) D → T L_V	Tipo = T.tipo
2) D → Ɛ	
3) T_R → struct {D}	STS.push(nuevaTS()) STT.push(nuevaTT()) S.dir.push(dir) dir=0 dir = S.dir.pop() TS = STS.pop() TT = STT.pop() TS.TT= TT T.tipo = STT.getCima().append('struct', tam, TS)
4) T→B T_A	B = B.base T.tipo = T_A.tipo
5) B → ent	B.tipo = ent
5) B → real	B.tipo = real
5) B → dreal	B.tipo = dreal
5) B → car	B.tipo = car
5) B → sin	B.tipo = sin
6) T_A → [num] T_A1	Si num.tipo = ent Entonces Si num.dir > 0 Entonces T_A.tipo = TT.append("array", num.dir, T_A1.tipo) Sino Error("El indice debe ser mayor a 0") Fin Si Sino Error("El índice debe de ser entero") Fin Si
6) T_A → Ɛ	T_A.tipo = base
7) L_V → L_V1, id	Si !TS.existe(id) Entonces STS.getCima().append(id, dir, T, 'var',nulo, -1) dir ← dir + STT.getCima().getTam(Tipo) Sino Error("El id ya existe") Fin Si
7) L → id	Si !TS.existe(id) Entonces STS.getCima().append(id, dir, Tipo, 'var',nulo, -1) dir ← dir + STT.getCima().getTam(Tipo) Sino

	Error(“Ya fue declarado el id”) Fin Si
8) F → def T id (ARG) {DS} F	Si !STS.getCima().existe(id) Entonces STS.push(nuevaTS()) S.dir.push(dir) dir=0 L_R = nuevaLista() Si cmpRet(L_R, T.tipo) Entonces L =nuevaEtiqueta() backpatch(S.nextlist, L) F.codigo = etiqueta(id) || S.codigo || etiqueta(L) Sino Error(“El valor no corresponde al tipo de la función”) Fin Si STS.pop() dir = S_D.pop( ) Sino Error(“El id ya fue declarado”) Fin Si
8) F → Ɛ	
9) ARG → L_ARG	ARG.lista = L_ARG.lista ARG.num = L_ARG.num
9) ARG → Ɛ	ARG.lista = nulo ARG.num = 0
10) L_ARG → L_ARG1, T id ARG	L_ARG.lista = L_ARG1.lista L_ARG.lista.append(T.tipo) L_ARG.num = L_ARG1.num +1
10) L_ARG → T id ARG	L_ARG.lista = nuevaLista() L_ARG.lista.append(T.tipo) L_ARG.num = L_ARG1.num +1
11)ARG → T_ARG id	Si STS.getCima().getId(id)= -1 Entonces STS.getCima().addSym(id,tipo,dir,”var”) dir = dir + STT.getCima().getTam(T) Sino Error(“El identificador ya fue declarado”) Fin ARG.tipo = T_ARG.tipo
12)T_ARG → B P_A	B = B.tipo T_ARG.tipo = P_A.tipo
13) P_A → [ ] P_A1	P_A.tipo = STT.append(“array”,-,P_A1.tipo, null)
13) P_A → Ɛ	P_A.tipo = B
14) S → S1S2	L =nuevaEtiqueta() backpatch(S1.nextlist, L) S.nextlist = S2.nextlist S.codigo = S1.codigo || etiqueta(L) || S2.codigo

14)S → S1	
15) S → si (E_B) entonces S1 fin	L =nuevaEtiqueta() backpatch(E_B.truelist, L) S.nextlist= combinar(E_B.falselist, S1.nextlist) S.codigo = E_B.codigo || etiqueta(L) || S1.codigo
15) S → si (E_B) entonces S1 sino S2 fin	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(E_B.truelist, L1) backpatch(E_B.falselist, L2) S.nextlist = combinar(S1.nextlist, S2.nextlist) S.codigo = E_B.codigo || etiqueta(L1) || S1.codigo || gen('goto' S1.nextlist[0]) || etiqueta(L2) || S2.codigo
15) S → mientras (E_B) hacer S1 fin	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(S1.nextlist, L1) backpatch(B.truelist, L2) S.nextlist = B.falselist S.codigo = etiqueta(L1) ||B.codigo || etiqueta(L2) || S1.codigo || gen('goto' S1.nextlist[0]) || S2.codigo
15) S →segun (V) hacer CP fin	
15) S → V := E;	S.codigo = E.código || V || '='|| E.dir
15) S → escribir E;	S.codigo = gen("print") || E.codigo
15) S → leer V;	S.codigo=gen("scan" E.dir) S.listnext = nulo
15) S →devolver E;	S.nextlist = nulo L_R.append(E.tipo) S.codigo = gen(return E.dir)
15) S → devolver;	S.nextlist = nulo S.codigo = gen(return)
15) S → terminar;	L = nuevaEtiqueta() S.codigo=gen('goto' L) S.nextlist = nuevaLista() S.nextlist.add(L)
16) CASS → caso num: S CASS1	
16) CASS → caso num: S	
17) PRED → PRED: S	
17) PRED → Ɛ	
18) E_B → E_B1 || E_B2	L = nuevaEtiqueta() backpatch(E_B1.falselist, L) E_B.truelist = combinar(E_B1.truelist, E_B2.truelist) E_B.falselist = E_B2.falselist E_B.codigo = E_B1.codigo || etiqueta(L) || E_B2.codigo

18) E_B → E_B1 && E_B2	L = nuevaEtiqueta() backpatch(E_B1.truelist, L) E_B.truelist = E_B2.truelist E_B.falselist = combinar(E_B1.falselist, E_B2.falselist) E_B.codigo = E_B1.codigo || etiqueta(L) || E_B2.codigo
18) E_B → ! E_B1	B.truelist =B1.falselist B.falselist = B1.truelist B.codigo = B1.codigo
18) E_B → E1 R E2	t0 = nuevoIndice() t1 = nuevoIndice() B.truelist=crearLista(t0) B.falselist=crearLista(t1) B.codigo = gen('if' E1.dir R.op E2.dir 'goto' t0) || gen('goto' t1)
18) E_B →verdadero	t0 = nuevoIndice() E_B.truelist = nuevaLista(t0) E_B.codigo = gen('goto' t0)
18) C → falso	t0 = nuevoIndice() E_B.falselist = crearLista(t0) E_B.codigo = gen('goto' t0)
19) R → R1 < R2	R.dir = nuevaTemp R.tipo = maximo(R1.tipo , R2.tipo) t1= ampliar(R1.dir,R1.tipo,R.tipo) t2= ampliar(R2.dir, R2.tipo,R.tipo) R.codigo = gen(R.dir'=' t1'<'t2)
19) R → R1 > R2	R.dir = nuevaTemp R.tipo = maximo(R1.tipo , R2.tipo) t1= ampliar(R1.dir,R1.tipo,R.tipo) t2= ampliar(R2.dir, R2.tipo,R.tipo) R.codigo = gen(R.dir'=' t1'>'t2)
19) R → R1 >= R2	R.dir = nuevaTemp R.tipo = maximo(R1.tipo , R2.tipo) t1= ampliar(R1.dir,R1.tipo,R.tipo) t2= ampliar(R2.dir, R2.tipo,R.tipo) R.codigo = gen(R.dir'=' t1'>='t2)
19) R → R1 <= R2	R.dir = nuevaTemp R.tipo = maximo(R1.tipo , R2.tipo) t1= ampliar(R1.dir,R1.tipo,R.tipo) t2= ampliar(R2.dir, R2.tipo,R.tipo) R.codigo = gen(R.dir'=' t1'<='t2)
19) R → R1 <> R2	R.dir = nuevaTemp R.tipo = maximo(R1.tipo , R2.tipo) t1= ampliar(R1.dir,R1.tipo,R.tipo) t2= ampliar(R2.dir, R2.tipo,R.tipo) R.codigo = gen(R.dir'=' t1'<>'t2)
19) R → R1 = R2	R.dir = nuevaTemp

	R.tipo = maximo(R1.tipo , R2.tipo) t1= ampliar(R1.dir,R1.tipo,R.tipo) t2= ampliar(R2.dir, R2.tipo,R.tipo) R.codigo = gen(R.dir'=' t1'='t2)
19) R → E	R.dir =R.dir R.codigo =E.codigo
20) E → E1 + E2	E.tipo = maximo(E1.tipo, E2.tipo) E.dir = nuevaTemp() t1 = ampliar(E1.dir, E1.tipo, E.tipo) t2 = ampliar(E2.dir, E2.tipo, T.tipo) E.codigo = || E2.codigo || T.dir '=' t1 '+' t2
20) E → E1 – E2	E.tipo = maximo(E1.tipo, E2.tipo) E.dir = nuevaTemp() t1 = ampliar(E1.dir, E1.tipo, E.tipo) t2 = ampliar(E2.dir, E2.tipo, T.tipo) E.codigo = || E2.codigo || T.dir '=' t1 '-' t2
20) E → E1 * E2	E.tipo = maximo(E1.tipo, E2.tipo) E.dir = nuevaTemp() t1 = ampliar(E1.dir, E1.tipo, E.tipo) t2 = ampliar(E2.dir, E2.tipo, T.tipo) E.codigo = || E2.codigo || T.dir '=' t1 '*' t2
20) E → E1 / E2	E.tipo = maximo(E1.tipo, E2.tipo) E.dir = nuevaTemp() t1 = ampliar(E1.dir, E1.tipo, E.tipo) t2 = ampliar(E2.dir, E2.tipo, T.tipo) E.codigo = || E2.codigo || T.dir '=' t1 '/' t2
20) E → E1 % E2	E.tipo = maximo(E1.tipo, E2.tipo) E.dir = nuevaTemp() t1 = ampliar(E1.dir, E1.tipo, E.tipo) t2 = ampliar(E2.dir, E2.tipo, T.tipo) E.codigo = || E2.codigo || T.dir '=' alfa1 ' %' alfa2
20) E → (E1)	
20) E → V	Si TS.existe(variable) Entonces E.dir =V.dir E.tipo = TS.getTipo(V) Sino error("La variable no ha sido declarada") Fin Si
20) E → cadena	E.tipo = cadena E.dir =TOS.add(cadena)
20) E → num	E.tipo = num.tipo E.dir = num.val
20) E → car	E.tipo = car E.dir =TOS.add(car)
21) V → id V_C	
22) V_C → D_S_T	

22) V_C → A	V_C.dir =A.dir V_C.base = A.base V_C.tipo =A.tipo
22) V_C → (P)	V_C.lista =P.lista V.num = P.num
23) D_S_T → D_S_T.id	
23) D_S_T → Ɛ	
24) A → id [E]	A.dir = nuevaTemp() A.base = id A.tipo = TT.getTipoBase(id.tipo) A.codigo = E.codigo || A.dir '=' E.dir 'x' TT.getTam(A.tipo)
24) A → A1 [E]	A.base = A1.base A.tipo = TT.getTipoBase(A1.tipo) Temp = nuevaTemp() A.dir = nuevaTemp() A.codigo = A1.codigo || E.codigo || temp '=' E.dir 'x' TT.getTam(A.tipo) || A.dir '=' A1.dir '+' temp
25) P → L_P	P.lista =L_P.lista P.num =L_P.num
25) P → Ɛ	
26) L_P → L_P1, E	L_P.lista = L_P1.lista L_P.lista.append(E.tipo) L_P.num = L_P1.num +1