Glosario

programa → declaraciones funciones	$P \rightarrow D\ F$
declaraciones→ tipo lista_var **;** declaraciones	$D \rightarrow T\ L_v$ **;** D_1
declaraciones→ tipo registro lista_var **;** declaraciones	$D \rightarrow T\ R\ L_v$ **;** D_1
declaraciones→ ε	$D \rightarrow\ \varepsilon$
tipo registro → **estructura inicio** declaraciones **fin**	$T_R \rightarrow$ **estructura inicio** D **fin**
tipo → base tipo_arreglo	$T \rightarrow B\ T_A$
base → **ent**	$B \rightarrow$ **ent**
base → **real**	$B \rightarrow$ **real**
base → **dreal**	$B \rightarrow$ **dreal**
base → **car**	$B \rightarrow$ **car**
base → **sin**	$B \rightarrow$ **sin**
tipo_arreglo → **[num]** tipo arreglo	$T_A \rightarrow$ **[num]** T_{A1}
tipo_arreglo → ε	$T_A \rightarrow \varepsilon$
lista_var → lista_var **, id**	$L_V \rightarrow L_{V1}$ **, id**
lista_var → **id**	$L_V \rightarrow$ **id**
funciones → **def** tipo **id(**argumentos**) inicio** declaraciones sentencias **fin** funciones	$F \rightarrow$ **def** T **id(** A_s **) inicio** $D\ S_s$ **fin** F_1
funciones → ε	$F \rightarrow \varepsilon$
argumentos → lista_ arg	$A_s \rightarrow L_A$
argumentos →**sin**	$A_s \rightarrow$ **sin**
lista_arg → lista arg, arg	$L_A \rightarrow L_{A1}$ **,** A
lista_arg → arg	$L_A \rightarrow A$
arg → tipo_arg **id**	$A \rightarrow T_A$ **id**
tipo_arg → base param_arr	$T_A \rightarrow B\ P_A$
param_arr → **[ ]** param_arr	$P_A \rightarrow$ **[ ]** P_{A1}
param_arr →ε	$P_A \rightarrow \varepsilon$
sentencias → sentencias sentencia	$S_s \rightarrow S_{s1}\ S$
sentencias → sentencia	$S_s \rightarrow S$

sentencia → **si** e_bool **entonces** sentencia **fin**	S →**si** E_B **entonces** S_1 **fin**
sentencia → **si** e_bool **entonces** sentencia **sino** sentencia **fin**	S→ **si** E_B **entonces** S_1 **sino** S_2 **fin**
sentencia → **mientras** e_bool **hacer** sentencia **fin**	S → **mientras** E_B **hacer** S_1 **fin**
sentencia → **hacer** sentencia **mientras** e_bool **;**	S → **hacer** S_1 **mietras** e_bool **;**
sentencia → **segun (** variable **) hacer** casos predeterminado **fin**	S → **segun (** V **) hacer** C P_R **fin**
sentencia → variable **:=** expresion **;**	S → V **:=** E **;**
sentencia → **escribir** expresion **;**	S → **escribir** E **;**
sentencia → **leer** variable **;**	S → **leer** V **;**
sentencia → **devolver ;**	S → **devolver ;**
sentencia → **devolver** expresion **;**	S → **devolver** E **;**
sentencia → **terminar ;**	S → **terminar ;**
sentencia → **inicio** sentencias **fin**	S → **inicio** S_S **fin**
casos → **caso num:** sentencia casos	C → **caso num:** S C_1
casos → **caso num:** sentencia	C → **caso num:** S
predeterminado → **pred:** sentencia	P_R → **pred:** S
predeterminado → ε	P_R → ε
e_bool → e_bool **o** e_bool	E_B → E_{B1} **o** E_{B2}
e_bool → e_bool **y** e_bool	E_B → E_{B1} **y** E_{B2}
e_bool → **no** e_bool	E_B → **no** E_{B1}
e_bool → relacional	E_B → R_L
e_bool → **verdadero**	E_B → **verdadero**
e_bool → **falso**	E_B → **falso**
relacional → relacional **>** relacional	R_L→R_{L1} **>** R_{L2}
relacional → relacional **<** relacional	R_L → R_{L1} **<** R_{L2}

relacional → relacional **<=** relacional	$R_L \rightarrow R_{L1}$ **<=** R_{L2}
relacional → relacional **>=** relacional	$R_L \rightarrow R_{L1}$ **>=** R_{L2}
relacional → relacional **<>** relacional	$R_L \rightarrow R_{L1}$ **<>** R_{L2}
relacional → relacional **=** relacional	$R_L \rightarrow R_{L1}$ **=** R_{L2}
relacional → expresion	$R_L \rightarrow E$
expresion → expresion **+** expresion	$E \rightarrow E_1$ **+** E_2
expresion → expresion **−** expresion	$E \rightarrow E_1$ **-** E_2
expresion → expresion **∗** expresion	$E \rightarrow E_1$ **∗** E_2
expresion → expresion **/** expresion	$E \rightarrow E_1$ **/** E_2
expresion → expresion **%** expresion	$E \rightarrow E_1$ **%** E_2
expresion → **(** expresion **)**	$E \rightarrow$ **(** E_1 **)**
expresion → variable	$E \rightarrow V$
expresion → **num**	$E \rightarrow$ **num**
expresion →**cadena**	$E \rightarrow$ **cadena**
expresion → **caracter**	$E \rightarrow$ **caracter**
variable → **id** variable_comp	$V \rightarrow$ **id** V_C
variable_comp → dato_est_sim	$V_C \rightarrow D_{ES}$
variable_comp → arreglo	$V_C \rightarrow A_R$
variable_comp → **(** parametros **)**	$V_C \rightarrow$ **(** P_S **)**
dato_est_sim → dato_est_sim **.id**	$D_{ES} \rightarrow D_{ES1}$**.id**
dato_est_sim → ε	$D_{ES} \rightarrow$ ε
arreglo → **[** expresion **]**	$A_R \rightarrow$ **[**E**]**
arreglo → arreglo **[** expresion **]**	$A_R \rightarrow A_{R1}$ **[**E**]**
parametros → lista_param	$P_S \rightarrow L_P$
parametros → ε	$P_S \rightarrow$ ε
lista_param → lista_param **,** expresion	$L_P \rightarrow L_{P1}$ **,** E
lista_param → expresion	$L_P \rightarrow$E

Definición dirigida por sintaxis:

$P \rightarrow D\ F$	STS.push(nuevaTS()) STT.push(nuevaTT()) dir =0 P.code = F.code
$D \rightarrow T\ L_v$ **;** D_1	Tipo = T.tipo
$D \rightarrow T\ R\ L_v$ **;** D_1	Tipo = T_R.tipo
$D \rightarrow\ \varepsilon$	
$T_R \rightarrow$ **estructura inicio** D **fin**	STS.push(nuevaTS()) STT.push(nuevaTT()) Sdir.push(dir) dir=0 dir = Sdir.pop() Ts = STS.pop() Tt = STT.pop() ts.tt = Tt T.tipo = STT.getCima().append('estructura', tam, Ts)
$T \rightarrow B\ T_A$	B.tipo = B.base T.tipo = T_A.tipo
$B \rightarrow$ **ent**	B.base = ent
$B \rightarrow$ **real**	B.base = real
$B \rightarrow$ **dreal**	B.base = dreal
$B \rightarrow$ **car**	B.base = car
$B \rightarrow$ **sin**	B.base = sin

$T_A \rightarrow$ **[num]** T_{A1}	**Si** num.tipo = entero **Entonces**
	  Si num.dir > 0 **Entonces**
	  T_A.tipo = TT.apend("arreglo", num.dir, T_{A1}.tipo)
	  Sino
	  Error("...")
	  Fin Si
	Sino
	  Error("...")
	Fin Si

$T_A \rightarrow \varepsilon$	T_A.tipo = Base
$L_V \rightarrow L_{V1}$, **id**	Si !Ts.existe(id) Entonces STS.getCima().append(id, dir, Tipo, 'var',nulo, -1) Dir ← dir + STT.getCima().getTam(Tipo) Sino Error(...) Fin Si
$L_V \rightarrow$ **id**	Si !Ts.existe(id) Entonces

	STS.getCima().append(id, dir, Tipo, 'var',nulo, -1) Dir ← dir + STT.getCima().getTam(Tipo) Sino Error(...) Fin Si
F → **def** T **id(** A_s **)** **inicio** D S_s **fin** F_1	Si no STS.getCima().existe(id) Entonces STS.push(nuevaTS()) Sdir.push(dir) dir=0 lista_retorno = nuevaLista() Si cmpRet(lista_retorno, T.tipo) Entonces L =nuevaEtiqueta() backpatch(S.nextlist, L) F.code = etiqueta(id) || S.code || etiqueta(L) Sino Error("el valor no corresponde al tipo de la Función") Fin Si STS.pop() dir= Sdir.pop() Sino Error("El id ya fue declarado") Fin Si --Duda
F → ε	
A_s → L_A	A_s.lista = L_A.lista A_s.num = L_A.num
A_s → **sin**	A_s.lista = nulo A_s.num = 0
L_A → L_{A1} **,** A	L_A.lista = L_{A1}.lista L_A.lista.append(A.tipo) L_a.num = L_{A1}.num + 1
L_A → A	L_A.lista = nuevaLista() L_A.lista.append(A.tipo) L_A.num = 1
A → T_A **id**	
T_A → B P_A	B = B.base T_A.tipo = P_A.tipo
P_A → **[ ]** P_{A1}	P_A.tipo = STS.getCima.insertar("arreglo"), 0 , P_{A1}.tipo
P_A →ε	
S_s →S_{s1} S	L = nuevaEtiqueta() backpatch(S_{s1} .nextlist, L) S.code = S_{s1} .code || etiqueta (L) || S.code

$S_s \rightarrow S$	S.lista = nuevaLista() S.lista.append(S.tipo) S.num = 1
S →**si** E_B **entonces** S_1 **fin**	L = nuevaEtiqueta() backpatch(E_a.truelist,) S.nextlist = combinar(E_a.falselist, S_1.nextlist) S.code = E_a.code || etiqueta (L) || S_1.code
S→ **si** E_B **entonces** S_1 **sino** S_2 **fin**	L_1 = nuevaEtiqueta() L_2 = nuevaEtiqueta() backpatch(E_B.truelist, L_1) backpatch(E_B.falselist, L_2) S.nextlist = combinar(S_1.nextlist, S_2.nextlist) S.code = E_B.code || etiqueta(L_1) || S_1.code || gen('goto' S_1.nextlist[0]) || etiqueta(L_2) || S_2.code
S → **mientras** E_B **hacer** S_1 **fin**	L_1 = nuevaEtiqueta() L_2 = nuevaEtiqueta() backpatch(E_B.nextlist, L_1) backpatch(E_B.truelist, L_2) S.nextlist = E_a.falselist S.code = etiqueta (L_1) ||$E_a$code || etiqueta(L_2) ||$S_{1.}$code ||gen('goto'S_1.nextlist[0]) ||S_2.code
S → **hacer** S_1 **mietras** e_bool **;**	
S → **segun (** V **) hacer** C P_R **fin**	
S → V **:=** E **;**	S.nextlist = nulo Si TS.existe(V) Entonces tipo_id = TS.getTipo(V) t = reducir(E.dir, E.tipo, V.tipo) gen(id '=' t) Sino error("El id no ha sido declarado") Fin Si
S → **escribir** E **;**	S.nextlist = nulo lista.retorno.append(E.tipo) S.code = gen(return E.dir)
S → **leer** V **;**	S.nextlist = nulo lista_retorno.append(V.tipo) S.code = gen(return V.dir)
S → **devolver ;**	S.nextlist = nulo --Duda
S → **devolver** E **;**	S.nextlist = nulo lista_retorno.append(E.tipo)

	S.code = gen(return E.dir)
S → **terminar ;**	S.nextlist = nulo --Duda
S → **inicio** S_S **fin**	
C → **caso num:** S C_1	
C → **caso num:** S	
P_R → **pred:** S	
P_R → ε	
E_B → E_{B1} **o** 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.code = E_{B1}.code || etiqueta(L) || E_{B2}.code
E_B → E_{B1} **y** 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.code = E_{B1}.code || etiqueta(L) || E_{B2}.code
E_B → **no** E_{B1}	E_B.truelist = E_{B1}.falselist E_B.falselist = E_{B1} .truelist E_B.code = E_{B1}.code
E_B → R_L	
E_B → **verdadero**	t_0= nuevoIndice() E_B.truelist = crearlista(t_0) E_B.code = gen('goto' t_0)
E_B → **falso**	t 0 = nuevoIndice() E_B.falselist = crearlista(t_0) E_B.code = gen('goto' t_0)
R_L→R_{L1} **>** R_{L2}	t_0 = nuevoIndice() t_1 = nuevoIndice() R_L.truelist=crearLista(t_0) B.falselist=crearLista(t_1) R_L.code = gen('if' R_{L1} .dir **>** R_{L2} .dir 'goto' t_0) || gen('goto' t_1)
R_L → R_{L1} **<** R_{L2}	t_0 = nuevoIndice() t_1 = nuevoIndice() R_L.truelist=crearLista(t_0) B.falselist=crearLista(t_1) R_L.code = gen('if' R_{L1} .dir **<** R_{L2} .dir 'goto' t_0) || gen('goto' t_1)
R_L → R_{L1} **<=** R_{L2}	t_0 = nuevoIndice() t_1 = nuevoIndice()

	R_L.truelist=crearLista(t_0) B.falselist=crearLista(t_1) R_L.code = gen('if' R_{L1} .dir **<=** R_{L2} .dir 'goto' t_0) || gen('goto' t_1)
$R_L \rightarrow R_{L1}$ **>=** R_{L2}	t_0 = nuevoIndice() t_1 = nuevoIndice() R_L.truelist=crearLista(t_0) B.falselist=crearLista(t_1) R_L.code = gen('if' R_{L1} .dir **>=** R_{L2} .dir 'goto' t_0) || gen('goto' t_1)
$R_L \rightarrow R_{L1}$ **<>** R_{L2}	t_0 = nuevoIndice() t_1 = nuevoIndice() R_L.truelist=crearLista(t_0) B.falselist=crearLista(t_1) R_L.code = gen('if' R_{L1} .dir **<>** R_{L2} .dir 'goto' t_0) || gen('goto' t_1)
$R_L \rightarrow R_{L1}$ **=** R_{L2}	t_0 = nuevoIndice() t_1 = nuevoIndice() R_L.truelist=crearLista(t_0) B.falselist=crearLista(t_1) R_L.code = gen('if' R_{L1} .dir **=** R_{L2} .dir 'goto' t_0) || gen('goto' t_1)
$R_L \rightarrow E$	
$E \rightarrow E_1$ **+** E_2	E.dir = nuevaTemporal E.tipo = max(E_1.tipo , E_2.tipo) t_1 = ampliar(E_1.dir, E_1.tipo , E.tipo) t_2 = ampliar(E_2.dir, E_2 .tipo , E.tipo) E.code = gen(E.dir'=' t_1' + 't_2)
$E \rightarrow E_1$ **-** E_2	E.dir = nuevaTemporal E.tipo = max(E_1.tipo , E_2.tipo) t_1 = ampliar(E_1.dir, E_1.tipo , E.tipo) t_2 = ampliar(E_2.dir, E_2 .tipo , E.tipo) E.code = gen(E.dir'=' t_1' - 't_2)
$E \rightarrow E_1$ ***** E_2	E.dir = nuevaTemporal E.tipo = max(E_1.tipo , E_2.tipo) t_1 = ampliar(E_1.dir, E_1.tipo , E.tipo) t_2 = ampliar(E_2.dir, E_2 .tipo , E.tipo) E.code = gen(E.dir'=' t_1' * 't_2)
$E \rightarrow E_1$ **/** E_2	E.dir = nuevaTemporal E.tipo = max(E_1.tipo , E_2.tipo) t_1 = ampliar(E_1.dir, E_1.tipo , E.tipo) t_2 = ampliar(E_2.dir, E_2 .tipo , E.tipo) E.code = gen(E.dir'=' t_1' / 't_2)
$E \rightarrow E_1$ **%** E_2	E.dir = nuevaTemporal E.tipo = max(E_1.tipo , E_2.tipo) t_1 = ampliar(E_1.dir, E_1.tipo , E.tipo) t_2 = ampliar(E_2.dir, E_2 .tipo , E.tipo)

	E.code = gen(E.dir'=' t_1' % 't $_2$)
E → **(** E_1 **)**	
E → V	
E → **num**	E.base --Duda
E → **cadena**	E.base = cadena
E → **caracter**	E.base = car
V → **id** V_C	
V_C → D_{ES}	
V_C → A_R	
V_C → **(** P_S **)**	
D_{ES} → D_{ES1}**.id**	
D_{ES} → ε	
A_R → **[**E**]**	
A_R → A_{R1} **[**E**]**	Si TS.existe(id) Entonces tipo_id = TS.getTipo(id) Si TT.getNombre(tipo_id) = array Entonces A_R .dir = nuevaTemporal() A_R .tipo = TT.getTipoBase(tipo_id) A_R .tam = TT.getTam(A_R.tipo) A_R .base = id.dir A_R .code = gen(A_R .dir '=' E.dir '*' A_R .tam) Sino error("El id no es un arreglo") Fin Si Sino error("El id no ha sido declarado") Fin Si --Duda
P_S → L_P	P_S.lista = L_p .lista P_S.num = L_p .num
P_S→ ε	P_S.lista = nulo P_S.num =0
L_P→L_{P1} **,** E	L_p.lista = L_{p1} .lista L_p.lista.append(E.tipo) L_p.num = L_{p1} .num +1
L_P →E	L_p.lista = nuevaLista() L_p.lista.append(E.tipo) L_p.num = 1