|  |  |
| --- | --- |
| **PRODUCCIÓN** | **REGLA SEMÁNTICAS** |
| programa → declaraciones funciones  P→ D F | STS.push(nuevaTS())  STT.push(nuevaTT())  dir =0  P.code = S.code |
| declaraciones → tipo lista\_var; declaraciones | tipo\_registro lista\_var;declaraciones  D→ T Lv ; | Tipo = T.tipo |
| declaraciones → ε \* | declaraciones.tipo |
| tipo\_registro → estructura inicio declaraciones fin  T→ struct{D} | STS.push(nuevaTS())  STT.push(nuevaTT())  Sdir.push(dir)  dir=0  dir = Sdir.pop()  Ts = STS.pop()  Tt = STT.pop()  ts.tt = Tt  T.ipo = STT.getCima().append(‘struct’, tam, Ts) |
| tipo → base tipo\_arreglo  T→ TB C | Base = TB .base  T.tipo = C.tipo |
| base → ent  TB→ ent | TB .base = int |
| base → real  TB → real | TB .base = float |
| base → dreal  TB → dreal | TB .base = double |
| base → car  TB → car | TB .base = char |
| base → sin  TB → sin | TB .base = void |
| tipo arreglo → [num] tipo\_arreglo  C→ [num] C1 | **Si** num.tipo = entero **Entonces**  **Si** num.dir > 0 Entonces C.tipo =  TT.append(“array”, num.dir, C1 .tipo)  **Sino**  Error(“…”)  **Fin Si**  **Sino**  Error(“…”)  **Fin Si** |
| tipo\_arreglo → ε | tipo\_arreglo.tipo = base |
| lista\_var → lista\_var, id | id  Lv → Lv1 , 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** |
| lista\_var → id  Lv → 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** |
| funciones → def tipo id(argumentos) inicio declaraciones sentencias fin funciones | ε  F→define T id (A) { D S } | **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** |
| argumentos → listar\_arg  A→ La | A.lista = La .lista  A.num = La .num |
| argumentos → sin  A→ ε | A.lista = nulo  A.num =0 |
| lista\_arg → lista\_arg, arg | arg  La → La1, A |  |
| lista\_arg → arg  La → A |  |
| arg → tipo\_arg id |  |
| tipo arg → base param arr |  |
| param arr → [ ] param arr |  param arr → ε |  |
| sentencias → sentencias sentencia | sentencia  S→ S1 S2 | L =nuevaEtiqueta()  backpatch(S1 .nextlist, L)  S.nextlist = S2 .nextlist  S.code = S1 .code ||etiqueta(L) || S2 .code |
| sentencia → si e bool entonces sentencia fin  S→ if(B) S1 | L =nuevaEtiqueta()  backpatch(B.truelist, )  S.nextlist= combinar(B.falselist, S1 .nextlist) S.code = B.code || etiqueta(L) || S1 .code |
| sentencia → si e\_bool entonces sentencia sino sentencia fin  S→if(B) S1 else S2 | L1 = nuevaEtiqueta()  L2 = nuevaEtiqueta()  backpatch(B.truelist, L1 ) backpatch(B.falselist, L2 )  S.nextlist = combinar(S1 .nextlist, S2 .nextlist) S.code = B.code || etiqueta(L1 ) || S1 .code  || gen(‘goto’ S1 .nextlist[0])  || etiqueta(L2 ) || S2 .code |
| sentencia → mientras e bool hacer sentencia fin | L1 = nuevaEtiqueta()  L2 = nuevaEtiqueta()  backpatch(S1 .nextlist, L1 ) backpatch(B.truelist, L2 )  S.nextlist = B.falselist  S.code = etiqueta(L1 )  ||B.code || etiqueta(L2 )  || S1 .code || gen(‘goto’ S1 .nextlist[0])  || S2 .code |
| sentencia → inicio sentencias fin  S→ {S1 } | S.nextlist = S1 .nextlist |
| e\_bool → e\_bool o e\_bool  B→ B1 || B2 | L = nuevaEtiqueta()  backpatch(B1 .falselist, L )  B.truelist = combinar(B1 .truelist, B2 .truelist) B.falselist = B2 .falselist  B.code = B1 .code || etiqueta(L) || B2 .code |
| e\_bool → e\_bool y e\_bool  B→ B1 && B2 | L = nuevaEtiqueta()  backpatch(B1 .truelist, L)  B.truelist = B2 .truelist  B.falselist = combinar(B1 .falselist, B2 .falselist)  B.code = B1 .code || etiqueta(L) || B2 .code |
| B→ !B1  e\_bool → no e\_bool | B.truelist =B1 .falselist  B.falselist = B1 .truelist  B.code = B1 .code |
| B→ relacional  B→ E1 rel E2 | B.truelist = relacional.truelist  B.falselist = relacional.falseliste |
| B→ verdadero  B→ true | t0 = nuevoIndice()  B.truelist = crearlista(t0 )  B.code = gen(‘goto’ t0 ) |
| B→ falso  B→ false | t0 = nuevoIndice()  B.falselist = crearlista(t0 )  B.code = gen(‘goto’ t0 ) |
| relacional → relacional1 > relacional2l | t0 = nuevoIndice()  t1 = nuevoIndice()  relacional.truelist=crearLista(t0) relacional.falselist=crearLista(t1 )  relacional..code = gen(‘if’ relcional1 .dir “>” relacional2 .dir ‘goto’ t0 ) || gen(‘goto’ t1 ) |
| relacional → relacional1 < relacional2 | t0 = nuevoIndice()  t1 = nuevoIndice()  relacional.truelist=crearLista(t0) relacional.falselist=crearLista(t1 )  relacional..code = gen(‘if’ relcional1 .dir “<” relacional2 .dir ‘goto’ t0 ) || gen(‘goto’ t1 ) |
| relacional → relacional1 <= relacional2 | t0 = nuevoIndice()  t1 = nuevoIndice()  relacional.truelist=crearLista(t0) relacional.falselist=crearLista(t1 )  relacional..code = gen(‘if’ relcional1 .dir “<=” relacional2 .dir ‘goto’ t0 ) || gen(‘goto’ t1 ) |
| relacional → relacional1 >= relacional2 | t0 = nuevoIndice()  t1 = nuevoIndice()  relacional.truelist=crearLista(t0) relacional.falselist=crearLista(t1 )  relacional..code = gen(‘if’ relcional1 .dir “>=” relacional2 .dir ‘goto’ t0 ) || gen(‘goto’ t1 ) |
| relacional → relacional1 <> relacional2 | t0 = nuevoIndice()  t1 = nuevoIndice()  relacional.truelist=crearLista(t0) relacional.falselist=crearLista(t1 )  relacional..code = gen(‘if’ relcional1 .dir “<>” relacional2 .dir ‘goto’ t0 ) || gen(‘goto’ t1 ) |
| relacional → relacional1 = relacional2 | t0 = nuevoIndice()  t1 = nuevoIndice()  relacional.truelist=crearLista(t0) relacional.falselist=crearLista(t1 )  relacional..code = gen(‘if’ relcional1 .dir “=” relacional2 .dir ‘goto’ t0 ) || gen(‘goto’ t1 ) |
| relacional → expresion | relacional.dir = expresion.dir  relacional.tipo = exorexion.tipo |
| expresion → expresion1 + expresion2  E→ E1 + E2 | E.dir = nuevaTemporal  E.tipo = max(E1 .tipo , E2 .tipo)  t1 = ampliar(E1 .dir, E1 .tipo , E.tipo)  t2 = ampliar(E2 .dir, E2 .tipo , E.tipo)  E.code = gen(E.dir’=’ t1 ’+’t2 ) |
| expresion → expresion1 \* expresion2  E→ E1 \* E2 | E.dir = nuevaTemporal  E.tipo = max(E1 .tipo , E2 .tipo)  t1 = ampliar(E1 .dir, E1 .tipo , E.tipo)  t2 = ampliar(E2 .dir, E2 .tipo , E.tipo)  E.code = gen(E.dir’=’ t1 ’\*’t2 ) |
| expresion → expresion1 - expresion2  E→ E1 - E2 | E.dir = nuevaTemporal  inst = genera codig(E.dir ’=’ E1.dir ’-’ E2.dir)  **si** E1.first != -1 **entonces**  E.first = E1.first  **sino**  **si** E2.first != -1 **entonces**  E.first = E2.first  **sino**  E.first = inst  **ﬁn si**  **ﬁn si** |
| expresion → expresion1 / expresion2  E→ E1 / E2 | E.dir = nuevaTemporal  inst = genera codig(E.dir ’=’ E1.dir ’/’ E2.dir)  **si** E1.first != -1 **entonces**  E.first = E1.first  **sino**  **s**i E2.first != -1 **entonces**  E.first = E2.first  **sino**  E.first = inst  **ﬁn si**  **ﬁn si** |
| expresion → | expresion1 % expresion2 | E.dir = nuevaTemporal  si (E1.tipo == int && E2.tipo == int)  inst = genera codig(E.dir ’=’ E1.dir ’%’ E2.dir)  sino  error() |
| parametros → lista param  Pa → Lp | Pa .lista = Lp .lista  Pa .num = Lp .num |
| parametros → ε  Pa → ε | Pa .lista = nulo  Pa .num = 0 |
| lista param → lista param, expresion  Lp → Lp1, E | Lp .lista = Lp1.lista  Lp .lista.append(E.tipo)  Lp .num = Lp1.num +1 |
| lista param → expresion  L p → E | Lp .lista = nuevaLista()  Lp .lista.append(E.tipo)  Lp .num = 1 |
| sentencia →  | hacer sentencia mientras e bool; do while();  | segun (variable) hacer casos predeterminado fin  | variable := expresion ;  | escribir expresion ;  | leer variable ;  | devolver;  | devolver expresion;  | terminar; |  |
|  |  |
|  |  |

Faltantes:

casos → caso num: sentencia casos | caso num: sentencia

predeterminado → pred: sentencia | ε

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

variable → id variable comp

variable comp → dato est sim | arreglo | ( parametros )

dato est sim → dato est sim .id | ε

arreglo → [ expresion ] | arreglo [ expresion ] ya la tenemos (buscar)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_