| PRODUCCIÓN | REGLAS SEMÁNTICAS |
|---|---|
| programa → declaraciones funciones | PilaTS = PTS_nueva()<br>PilaTT = PTT_nueva()<br>PTS_push( PilaTS , TS_nueva( “Global” ) )<br>PTT_push( PilaTT , TT_nueva( “Global” ) )<br>programa .code = funciones.code<br>PTS_eliminar(PilaTS)<br>PTT_eliminar(PilaTT) |
| declaraciones → tipo lista_var PYC declaraciones1 | TipoG = tipo.tipo |
| declaraciones → tipo_registro lista_var PYC declaraciones | TipoG = tipo_registro .tipo |
| declaraciones → $\varepsilon$ | |
| tipo → base tipo_arreglo | base = base.base<br>tipo.tipo = tipo_arreglo.tipo |
| base → ent | base.tipo = ent |
| base → real | base.tipo = real |
| base → dreal | base.tipo = deal |
| base → car | base.tipo = car |
| base → sin | base.tipo = sin |
| tipo_registro →ESTRUCTURA INICIO declaraciones FIN | PTS_push( PilaTS , TS_nueva( “Estructura” ) )<br>PTT_push( PilaTT , TT_nueva( “Estructura” ) )<br>Tts = PTS_pop(PilaTS)<br>Ttt = PTT_pop(PilaTT)<br>Tts.tt = Ttt<br>tipo_registro.tipo = TT_nuevoRegistro(getCima_PTT(PilaTT),T_nuevo(“estructura”, Tts.dirMax,-1, Tts))<br><br>//dirMax es un atributo de la tabla de símbolos que contiene el tamaño de la tabla |
| tipo_arreglo→ CORIZ NUM CORDER tipo_arreglo | **Si** num.tipo = 0 **Entonces**<br>  **Si** num.dir > 0 **Entonces**<br>  tipo_arreglo.tipo = TT_nuevoRegistro(getCima_PTT(PilaTT),T_nuevo(“array”, num.dir, tipo_arreglo.tipo,NULL))<br>  **Sino**<br>  Error(“El tamaño del arreglo no es válido”)<br>  **Fin Si**<br>**Sino**<br>  Error(“El tamaño del arreglo no es un número entero”)<br>**Fin Si** |
| tipo_arreglo→ $\varepsilon$ | tipo_arreglo.tipo = base |

| | |
|---|---|
| lista_var → ID A | **Si** existeID(getCima_PTS(PilaTS), ID.dir) == -1 **Entonces**<br>TS_nuevoRegistro(getCima_PTT(PilaTT),getCima_PTS(PilaTS),S_nuevo(ID.dir, TipoG,"var", NULL))<br>**Sino**<br>Error("Ya fue declarada el identificador")<br>**Fin Si** |
| A → COMA ID A | **Si** existeID(getCima_PTS(PilaTS), ID.dir) == -1 **Entonces**<br>TS_nuevoRegistro(getCima_PTT(PilaTT),getCima_PTS(PilaTS),S_nuevo(ID.dir, TipoG,"var", NULL))<br>**Sino**<br>Error("Ya fue declarada el identificador")<br>**Fin Si** |
| A→ $\varepsilon$ | |

| | |
|---|---|
| funciones → DEF tipo ID LPAR argumentos RPAR INICIO declaraciones sentencias FIN funciones1 | **Si** !existeID(getCima_PTS(PilaTS), ID.dir) **Entonces**<br>PTS_push( PilaTS , TS_nueva(ID.dir ) )<br>lista_retorno = nuevaLista()<br>**Si** cmpRet(lista_retorno, tipo.tipo) **Entonces**<br>L =nuevaEtiqueta()<br>backpatch(sentencias.nextlist, L )<br>funciones.code = etiqueta(ID.dir) \|\| sentencias.code \|\| etiqueta(L) \|\| funiones1.code<br>**Sino**<br>Error( "el valor no corresponde al tipo de la Función")<br>**Fin Si**<br>PTS_pop(PilaTS)<br>TS_nuevoRegistro(getCima_PTT(PilaTT), getCima_PTS(PilaTS), S_nuevo(ID.dir,tipo.tipo,"funcion",argumentos.lista))<br>**Sino**<br>Error("El id ya fue declarado")<br>**Fin Si** |
| funciones → $\varepsilon$ | funciones.code = gen("") |
| lista_arg → lista_arg1 COMA arg | lista_arg.lista = lista_arg1.lista<br>agregarArg(lista_arg.lista, arg.tipo) |
| lista_arg→arg | lista_arg.lista = crearListaArg()<br>agregarArg(lista_arg.lista, arg.tipo) |
| argumentos → lista_arg | argumentos.lista = lista_arg.lista<br>argumentos.num = lista_arg.num |
| argumentos → $\varepsilon$ | argumentos.lista = NULL<br>argumentos.num = 0 |
| arg→tipo_arg ID | arg.tipo = tipo_arg.tipo<br>TS_nuevoRegistro(getCima_PTT(PilaTT), getCima_PTS(PilaTS), S_nuevo(ID.dir, tipo_arg.tipo,"var",NULL)) |

| | |
|---|---|
| sentencias → sentencias1 sentencia | L =nuevaEtiqueta()<br>backpatch(sentencias1 .nextlist, L)<br>sentencias.nextlist = sentencia.nextlist<br>sentencias.code = sentencias1.code \|\|etiqueta(L) \|\| sentencia.code |
| sentencias → $\varepsilon$ | sentencias.nextlits = NULL<br>funciones.code = gen(“”) |
| sentencia → SI e_bool ENTONCES sentencia1 FIN | L =nuevaEtiqueta()<br>backpatch(e_bool.truelist,L)<br>sentencia.nextlist= combinar(e_bool .falselist, sentencia1.nextlist)<br>sentencia.code = e_bool.code \|\| etiqueta(L) \|\| sentencia1$_1$.code |

| | |
|---|---|
| sentencia → SI e_bool ENTONCES sentencia1 SINO sentencia2 FIN | L = nuevaEtiqueta()<br>L2= nuevaEtiqueta()<br>backpatch(e_bool.truelist,L)<br>backpatch(e_bool.falselist,L2)<br>sentencia.nextlist = combinar(sentencia1.nextlist, sentencia2.nextlist)<br>sentencia.code = e_bool.code \|\| etiqueta(L) \|\| sentencia1.code<br>\|\| gen(“goto”) \|\| gen(getPrimero(sentencia1.nextlist))<br>\|\| etiqueta(L2) \|\| sentencia2.code |
| sentencia → MIENTRAS e_bool HACER sentencia1 FIN | L = nuevaEtiqueta()<br>L2= nuevaEtiqueta()<br>backpatch(sentencia1 .nextlist,L)<br>backpatch(e_bool.truelist,L2)<br>sentencia.nextlist = e_bool.falselist<br>sentencia.code = etiqueta(L)<br>\|\|e_bool.code \|\| etiqueta(L2)<br>\|\| sentencia1.code<br>\|\| gen(“goto”) \|\| gen(getPrimero(sentencia1.nextlist)) |
| sentencia → HACER sentencia1 MIENTRAS e_bool PYC | L = nuevaEtiqueta()<br>backpatch(e_bool.truelist,L)<br>sentencia.nextlist = e_bool.falselist<br>sentencia.code = etiqueta(L) \|\| sentencia1.code \|\| e_bool.code |
| sentencia → SEGUN LPAR variable RPAR HACER casos predeterminado FIN | switch = variable.dir<br>//switch es una variable global para almacenar la variable<br>sentencia.nextList = casos.nextList<br>combinar(sentencia.nextlist,predeterminado.nextlist)<br>sentencia.code = casos.code \|\| predefinido.code |

| | |
|---|---|
| sentencia → INICIO sentencias FIN | sentencia.nextlist = sentencias.nextlist<br>sentencia.code = sentencias.code |
| sentencia → variable ASIG expresion PYC | sentencia.nextlist = NULL<br>t = reducir(expresion.dir, expresion.tipo,variable.tipo)<br>sentencia.code = variable.dir = expresion.dir |
| sentencia → ESCRIBIR expresion PYC | sentencia.code = expresion.code \|\| gen("ESCRIBIR ") \|\| expresion.dir<br>sentencia.nextList = NULL |
| sentencia → LEER variable PYC | sentencia.code = gen("LEER ") \|\| variable.dir<br>sentencia.nextList = NULL |
| sentencia → DEVOLVER PYC | sentencia.nextList = NULL<br>sentencia.code = gen("return ") |
| sentencia → TERMINAR PYC | L = nuevaTemporal()<br>sentencia.nextList = crearLista()<br>backpatch(sentencia.nextList,L)<br>sentencia.code = gen("goto ") \|\| L |
| sentencia → DEVOLVER expresion PYC | sentencia.nextList = NULL<br>agregarArg(lista_retorno, expresion.tipo)<br>sentencia.code = expresion.code \|\| gen("return ") \|\| expresion.dir |
| casos→CASO NUM DOSP sentencias casos1 | L = nuevaEtiqueta()<br>L2 = nuevaEtiqueta()<br>backpatch(sentencia.nextlist,L)<br>casos.nextList = casos1.nextList<br>combinar(casos.nextlist,sentencias.nextlist)<br>casos.code =gen("if ")\|\| switch \|\| gen(" = ") \|\| NUM.dir<br>\|\| gen("goto ") \|\| gen(getPrimero(sentencia.nextlist))<br>\|\| gen("goto ") \|\| gen(L2)<br>\|\| etiqueta(L) \|\| sentencia.code<br>\|\| etiqueta(L2) \|\| casos.code |
| casos→CASO NUM DOSP sentencia | L = nuevaEtiqueta()<br>backpatch(sentencia.nextList,L)<br>casos.nextlist = sentencias.nextlist<br>casos.code = gen("if ")\|\| switch \|\| gen(" = ") \|\| NUM.dir<br>\|\| gen("goto ") \|\| gen(L)<br>\|\| gen("goto ") \|\| gen(getPrimero(sentencia.nextlist))<br>\|\| sentencia.code |
| predeterminado → PRED DOSP sentencia | predeterminado.nextlist = sentencias.nextlist<br>predeterminado.code = sentencia.code |
| predeterminado → $\varepsilon$ | predeterminado.nexlist = NULL<br>predetermindo.code = "" |

| | |
|---|---|
| expresion → expresion1 SUM_RES expresion2 | expresion.tipo = max(expresion1.tipo , expresion2.tipo)<br>t1= ampliar(expresion1.dir, expresion1.tipo , expresion.tipo)<br>t2= ampliar(expresion2.dir, expresion2.tipo , expresion.tipo)<br>expresion.dir = nuevaTemporal()<br>expresion.code = expresion.dir \|\| gen(" = ")<br>\|\| t1 \|\| SUM_RES.dir \|\| t2 |

| | |
|---|---|
| expresion → expresion1 MUL_DIV_MOD expresion2 | expresion.tipo = max(expresion1.tipo , expresion2.tipo)<br>t1= ampliar(expresion1.dir, expresion1.tipo , expresion.tipo) t2=<br>ampliar(expresion2.dir, expresion2.tipo , expresion.tipo)<br>expresion.dir = nuevaTemporal()<br>expresion.code = expresion.dir \|\| gen(" = ")<br>\|\| t1 \|\| MUL_DIV_MOD .dir \|\| t2 |
| expresion → variable | expresion.dir = varible.dir<br>expresion.tipo = variable.tipo |
| expresion→ LPAR expresion1 RPAR | expresion.dir = expresion1.dir<br>expresion.tipo = expresion1 .tipo |
| expresion→ NUM | expresion.dir = NUM.dir<br>expresion.tipo = ent |
| expresion→ CADENA | |
| expresion→ CARACTER | t = nuevaTemporal()<br>expresion.dir = t<br>expresion.tipo = car<br>expresion.code = t \|\| gen(" = ") \|\| ' CARACTER.dir' |
| variable→ ID variable_comp | **Si** existeID(getCima_PTS(PilaTS), ID.dir) **Entonces**<br>PID_push(PilaID,ID.dir))<br>variable.tipo = variable_comp.tipo<br>**Si** getTipo(getCima_PTS(PilaTS),ID.dir) = variable_comp.tipo **entonces**<br>nombre = getVar_TS(getCima_PTS(PilaTS),ID.dir)<br>**Si** nombre = "funcion" **entonces**<br>varible.dir = varible_comp.dir<br>variable.code = varible_comp.code<br>**Sino**<br>variable.dir = ID.dir<br>variable.code = ""<br>**FinSi**<br>**Sino**<br>t = nuevaTemporal()<br>variable.dir = t<br>variable.code = t \|\| gen(" = ")<br>\|\| ID.dir[variable_comp.dir]<br>**FinSi**<br>PID_pop(PilaID)<br>**Sino**<br>error("El id no ha sido declarado")<br>**Fin Si** |
| variable_comp→ dato_est_sim | variable_comp.tipo = dato_est_sim.base<br>variable_comp.code = dato_est_sim.code |
| variable_comp→arreglo | variable_comp.dir = arreglo.dir<br>variable_comp.tipo = arreglo.base |
| variable_comp→LPAR parametros RPAR | tmpLista = getArgs(getCima_PTS(PilaTS) , getCima_PID(PilaID))<br>**Si** compararListas(tmpLista , parametros.lista) != -1 **entonces**<br>tmpT = getTipo(getCima_PTS(PilaTS) , getCima_PID(PilaID))<br>variable_comp.tipo = tmpT<br>varable_comp.dir = nuevaTemporal()<br>variable_comp.code = parametros.code \|\| variable.dir \|\| gen(" = CALL ")<br>\|\| getCima_PID(PilaID)<br>\|\|gen( " , ") \|\| getTamlistaArg(ltmpLista) |

| | |
|---|---|
| | **sino**<br>Error("Los parametros de la funcion no estan bien definidos")<br>**FinSi** |
| parametros→ lista_param | parametros.lista = lista_param.lista<br>parametros.code = lista.param.code |
| parametros → $\varepsilon$ | parametros.lista = NULL |
| dato_est_sim → dato_est_sim PUNTO ID | nombre = getNombre_TT(getCima_PTT(PilaTT),dato_est_sim.base)<br>**Si** nombre = "Estructura" **entonces**<br>t0 = nuevaTemporal()<br>t1 = nuevaTemporal()<br><br>tmpTT = getCima_PTT(PilaTT)<br>tmpT = dato_est_sim.base<br>PTS_push(PilaTS, getTS(tmpTT, tmpT))<br>PTT_push(PilaTT, getTT(getCima_PTS(PilaTS)) )<br><br>direccion = getDir(getCima_PTS(PilaTS) , ID.dir)<br><br>dato_est_sim.code = t0 \|\| gen(" = ") \|\| direccion<br>t1 \|\| gen(" = ") \|\| dato_est_sim.dir \|\| gen(" + ") \|\| t0<br><br>dato_est_sim.base = getTipo( getCima_PTS(PilaTS),ID.dir )<br>dato_est_sim.dir = t1<br><br>PTS_pop()<br>PTT_pop()<br>**SIno**<br>Error("La variable asociado no corresponde a una estructura")<br>**FinSi** |
| dato_est_sim→ $\varepsilon$ | tmpT = getTipo(getCima_PTS(PilaTS) , getCima_PID(PilaID))<br>nombre = getNombre_TT(getCima_PTT(PilaTT),tmpT)<br>**Si** nombre = "Estructura" **entonces**<br>t0 = nuevaTemporal()<br>direccion = getDir(getCima_PTS(PilaTS) ,getCima_PID(PilaID)r)<br>dato_est_sim.code = t0 \|\| gen(" = ") \|\| direccion<br>**FinSi**<br>dato_est_sim.base = getTipo(getCima_PTS(PilaTS),getCima_PID(PilaID)) |
| arreglo→CORIZQ expresion CORDER arreglo1 | t = nuevaTemporal()<br>t2 = nuevaTemporal()<br>arreglo.dir = t2<br>arreglo.base = getBase_TT(getCima_PTT(PilaTT),arreglo1.base)<br>arreglo.code = t \|\| gen("=") \|\| expresion.dir \|\| gen("*")<br>\|\| TT_getTam(getCima_PTT(PilaTT),arreglo1.base)<br>\|\|t2 \|\| gen("=") \|\| arrglo1.dir \|\| gen("+") \|\| t |
| arreglo → CORIZQ expresion CORDER | t = nuevaTemporal()<br>arreglo.dir = t<br>tmpT = getTipo(getCima_PTS(PilaTS),getCima_PID(PilaID))<br>arreglo.base = getBase_TT(getCima_PTT(PilaTT),tmpT)<br>arreglo.code =expresion.code \|\|<br>t \|\| gen("=") \|\| expresion.dir \|\| gen("*")<br>\|\| TT_getTam(getCima_PTT(PilaTT),tmpT) |

| | |
|---|---|
| param_arr → CORIZQ CORDER param_arr1 | param_arr.tipo =TT_nuevoRegistro(getCima_PTT(PilaTT),T_nuevo("array", 0, param_arr1.tipo,NULL)) |

| | |
|---|---|
| param_arr⟶ $\varepsilon$ | param_arr.tipo = base |
| tipo_arg⟶base param_arr | base = base.base<br>tipo_arg.tipo = param_arr.tipo |
| parametros ⟶ $\varepsilon$ | parametros.lista = nulo |
| parametros ⟶ lista_param | parametros.lista = lista_param.lista<br>parametros.code = lista_param.code |
| lista_param ⟶ lista_param1 COMA expresion | lista_param.lista = lista_param1.lista<br>agregarArg(lista_param.lista, expresion.tipo)<br>lista_param.code = lista_param1.code \|\| expresion.code |
| lista_param ⟶ expresion | lista_param.lista = crearListaArg()<br>agregarArg(lista_param.lista, expresion.tipo)<br>lista_param.code = expresion.code |

| | |
|---|---|
| e_bool ⟶ e_bool1 O e_bool2 | L = nuevaEtiqueta()<br>backpatch(e_bool1.falselist, L )<br>e_bool.truelist = combinar(e_bool1.truelist, e_bool2.truelist)<br>e_bool.falselist = e_bool2.falselist<br>e_bool.code = e_bool1.code \|\| etiqueta(L) \|\| e_bool2.code |
| e_bool ⟶ e_bool1 Y e_bool2 | L = nuevaEtiqueta()<br>backpatch(e_bool1.truelist, L)<br>e_bool.truelist = e_bool2.truelist<br>e_bool.falselist = combinar(e_bool1.falselist, e_bool2.falselist)<br>e_bool.code = e_bool1.code \|\| etiqueta(L) \|\| e_bool2.code |
| e_bool ⟶ NO e_bool1 | e_bool.truelist =e_bool1.falselist<br>e_bool.falselist = e_bool1.truelist<br>e_bool.code = e_bool1.code |
| e_bool ⟶ relacional | e_bool.truelist=relacional.truelist<br>e_bool.falselist=relacional.falselist |

| | |
|---|---|
| e_bool → VERDADERO | t0 = nuevoIndice()<br>e_bool.truelist = crearlista()<br>agregarLista(e_bool.truelist,t0)<br>e_bool.code = gen("goto" ) \|\| t0 |
| e_bool → FALSO | t0 = nuevoIndice()<br>e_bool.falselist = crearLista(t0)<br>e_bool.code = gen("goto" ) \|\| t0 |
| relacional → relacional1 OPERADOR_RELACIONAL relacional2 | t0 = nuevoIndice()<br>t1 = nuevoIndice()<br>relacional.truelist=crearLista(t0)<br>relacional.falselist=crearLista(t1)<br>relacional.code = gen("if") \|\| relacional1.dir<br>\|\| OPERADOR_RELACIONAL.dir \|\| relacional2.dir<br>\|\| gen("goto ") \|\| t0<br>\|\| gen("goto") \|\| t1 |
| relacional → expresion | relacional.dir = expresion.dir<br>relacional.tipo = expresion.tipo<br>relacional.truelist = NULL<br>relacional.falselist = NULL |