

# Esquema de Traducción

---

Álvarez González Ian

García Oviedo Jaasiel Osmar

López López Ulysses

Domínguez Cisneros Alexis Saul

Gpo 1.

Compiladores |  
2020-2

## Glosario

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

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

## Esquema de traducción:

```
P → D {STS.push(nuevaTS())
        STT.push(nuevaTT())
        dir = 0
        P.codigo = F.codigo
    } F
D → TL_V {T = T.tipo
    } D1
D → T_R L_V D1 {T = T_R.tipo
    }
D → ε
T_R → estructura inicio D {STS.push(nuevaTS())
    STT.push(nuevaTT())
    SDir.push(dir)
    dir = 0
    dir = S.dir.pop()
    TS = STS.pop()
    TT = STT.pop()
    T_R.tipo = STT.getCima().append('struct', tam, TS)
    } Fin
T → B T_A {B = B.base
    T.tipo = T_A.tipo}
B → ent {STT.getCima().getTipo('ent')}
B → real {STT.getCima().getTipo('real')}
B → dreal {STT.getCima().getTipo('dreal')}
B → car {STT.getCima().getTipo('car')}
B → sin {STT.getCima().getTipo('sin')}
T_A → [num] {Si num.tipo = ent Entonces
    Si num.dir > 0 Entonces
        T_A.tipo = STT.getCima().append("array", num, T_A1.tipo)
    Sino
        Error("El indice debe ser mayor a 0")
    Fin Si
    Sino
        Error("El indice debe de ser entero")
    Fin Si
    } T_A1
T_A → ε {T_A.tipo = base}
L_V → L_V1, id {Si STS.getCima().existe(id) Entonces
    STS.getCima().append(id, dir, T, 'var', nulo, nulo)
    dir ← dir + STT.getCima().getTam(Tipo)
    Sino
        Error("El id ya existe")
    Fin Si
    }
```

---

```

L → id {Si STS.getCima.existe(id) Entonces
    STS.getCima().append(id, dir, Tipo, 'var', nulo, nulo)
    dir ← dir + STT.getCima().getTam(Tipo)
Sino
    Error("Ya fue declarado el id")
Fin Si
}

```

```

F → def T id {Si !STS.getGlobal().existe(id) Entonces
    STS.push(nuevaTS())
    STT.push(nuevaTT())
    SDir.push(dir)
    dir=0
    L_R = nuevaLista()
    Si cmpRet(L_R, T.tipo) Entonces
        L =nuevaEtiqueta()
        backpatch(S.nextlist, L )
        gen(etiqueta L)
        STS.pop()
        STT.pop()
    Sino
        Error( "El valor no corresponde al tipo de la función")
    Fin Si
Sino
    Error("El id ya fue declarado")
Fin Si
} (ARG) {DS} fin F

```

```

F → ε

```

```

ARG → L_ARG {ARG.lista = L_ARG.lista
    ARG.num = L_ARG.num}

```

```

ARG → sin {ARG.lista = nulo
    ARG.num = 0}

```

```

L_ARG → L_ARG1{L_ARG.lista = L_ARG1.lista
    L_ARG.lista.append(ARG.tipo)
    L_ARG.num = L_ARG1.num +1
} T id ARG

```

```

L_ARG → T id {L_ARG.lista = nuevaLista()
    L_ARG.lista.append(ARG.tipo)
    L_ARG.num = 1
} ARG

```

```

ARG → T_ARG id {Si ! STS.getCima().existe(id)= Entonces
    STS.getCima().append(id,T ,dir,"arg", nulo, nulo)
    dir ← dir + STT.getCima().getTam(T.tipo)
Sino
    Error("El identificador ya fue declarado")
Fin
}

```

```

T_ARG → B P_A {B = B.base
                T.tipo = P_A.tipo}
P_A → [ ] {P_A.tipo = STT.getCima().append("array",-P_A1.tipo) P_A1
P_A → ε {P_A.tipo = B}
S → S1{L =nuevaEtiqueta()
        backpatch(S1.nextlist, L)
        gen(etiqueta L)
    }S2
S → si (E_B) {L =nuevaEtiqueta()
              backpatch(E_B.truelist, L)
              S.nextlist= combinar(E_B.falselist, S1.nextlist)
              gen(etiqueta L)
    } entonces S1 fin
S → si (E_B) {L1 = nuevaEtiqueta()
              L2 = nuevaEtiqueta()
              backpatch(E_B.truelist, L1)
              backpatch(E_B.falselist, L2)
              S.nextlist = combinar(S1.nextlist, S2.nextlist)
              gen(etiqueta L1)
              gen('goto' S1.nextlist(0))
              gen(etiqueta L2)
    } entonces S1 sino S2 fin
S → mientras (E_B) {L1 = nuevaEtiqueta()
                    L2 = nuevaEtiqueta()
                    backpatch(S1.nextlist, L1)
                    backpatch(E_B.truelist, L2)
                    S.nextlist = E_B.falselist
                    gen(etiqueta L1)
                    gen(etiqueta L2)
                    gen('goto' S1.nextlist(0))
    } hacer S1 fin
S →segun (V) {L1 = nuevaEtiqueta()
              prueba = combinar(CASS.prueba,
                                PRED.prueba)
              backpatch(CASS.nextlist, L2)
              sustituir("??",V.dir, prueba)
    } hacer CP fin
S → V := E;{ dir = reducir(E.dir, E.tipo, V.tipo)
            Si V.codigo_est = true Entonces
                gen(V.base['variable.des'] '=' dir)
            Sino
                gen(V.dir '=' dir)
    Fin Si
}
S → escribir E; {gen("write" E.dir)}
S → leer V; {gen("read" E.dir)}
S →devolver E; { gen("return" E.dir)}

```

```

        index = nuevoIndice()
        S.nextlist = nuevaListaIndice(index)
        gen("goto" index)}
S → devolver; {gen('return')}
S → terminar; { gen("return" E.dir)
                index = nuevoIndice()
                S.nextlist = nuevaListaIndice(index)
                gen("goto" index)}
CASS → caso num: S CASS1 {CASS.nextlist = combinar(CASS.nextlist,
S1.nextlist)
                                L = nuevaEtiqueta()
                                /*Indica el inicio del codigo para la sentencia*/
                                gen("etiqueta" L)
                                CASS.prueba = casos1.prueba
                                CASS.prueba.append(if "???"
"=="num.dir"goto"L)
                                }
CASS → caso num: S{CASS.prueba = nuevoCodigo()
                    L = nuevaEtiqueta()
                    /*Indica el inicio del codigo para la sentencia*/
                    gen("etiqueta" L)
                    CASS.prueba.append(if "???" "==" num.dir "goto" L )
                    CASS.nextlist =S.nextlist
                    }
PRED → PRED: S {PRED.prueba = nuevoCodigo()
                L = nuevaEtiqueta()
                /*Indica el inicio del codigo para la sentencia*/
                gen("etiqueta" L)
                PRED.prueba.append("goto" L )
                }
PRED → ε {PRED.prueba = NULO}
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
                    gen(etiqueta L)}
E_B → E_B1 && E_B2 {L = nuevaEtiqueta()
                    backpatch(E_B1.truelist, L)
                    E_B.truelist = E_B1.truelist
                    E_B.falselist = combinar(E_B1.falselist,
E_B2.falselist)
                    gen(etiqueta L)}
E_B → ! E_B1{B.truelist =B1.falselist
              B.falselist = B1.truelist}
E_B → E1 R E2{E_B.truelist = R.truelist
              E_B.falselist = R.falselist}

```

---

```
E_B → verdadero {indice0 = nuevoIndice()
                  E_B.truelist = nuevaListaIndice(t0)
                  gen('goto' t0 )}
```

```
E_B → falso {indice0 = nuevoIndice()
              E_B.falselist = nuevaListaIndice(t0)
              gen('goto' t0 )}
```

```
R → R1 < R2 {indice0 = nuevoIndice()
              indice1 = nuevoIndice()
              R.truelist = nuevaListaIndice(indice0)
              R.falselist = nuevaListaIndice(indice1)
              gen('if' R1.dir < R2 'goto' indice0)
              gen('goto' indice1)}
```

```
R → R1 > R2 {indice0 = nuevoIndice()
              indice1 = nuevoIndice()
              R.truelist = nuevaListaIndice(indice0)
              R.falselist = nuevaListaIndice(indice1)
              gen('if' R1.dir > R2 'goto' indice0)
              gen('goto' indice1)}
```

```
R → R1 >= R2 {indice0 = nuevoIndice()
               indice1 = nuevoIndice()
               R.truelist = nuevaListaIndice(indice0)
               R.falselist = nuevaListaIndice(indice1)
               gen('if' R1.dir >= R2 'goto' indice0)
               gen('goto' indice1)}
```

```
R → R1 <= R2 {indice0 = nuevoIndice()
               indice1 = nuevoIndice()
               R.truelist = nuevaListaIndice(indice0)
               R.falselist = nuevaListaIndice(indice1)
               gen('if' R1.dir <= R2 'goto' indice0)
               gen('goto' indice1)}
```

```
R → R1 <> R2 {indice0 = nuevoIndice()
               indice1 = nuevoIndice()
               R.truelist = nuevaListaIndice(indice0)
               R.falselist = nuevaListaIndice(indice1)
               gen('if' R1.dir <> R2 'goto' indice0)
               gen('goto' indice1)}
```

```
R → R1 = R2 {indice0 = nuevoIndice()
              indice1 = nuevoIndice()
              R.truelist = nuevaListaIndice(indice0)
              R.falselist = nuevaListaIndice(indice1)
              gen('if' R1.dir = R2 'goto' indice0)
              gen('goto' indice1)}
```

```
R → E {R.dir =E.dir
        R.tipo =E.tipo }
```

```
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) gen(E.dir '=' t1 '+' t2)
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) gen(E.dir '=' t1 '-' t2)}
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) gen(E.dir '=' t1 '*' t2)}
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) gen(E.dir '=' t1 '/' t2)}
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) gen(E.dir '=' t1 '%' t2)}
E → (E1)	{E.tipo = E1.tipo E.dir = E1.dir}
E → V	{E.tipo = V.tipo E.dir = V.dir}
E → cadena	{E.tipo = 'string' Si TablaCadenas.existe(cadena) Entonces E.dir= TablaCadena.getIndiceStr(cadena) Sino E.dir=TablaCadena.append(cadena) Fin Si
	}
E → num	{E.tipo = num.tipo E.dir = num.dir}
E → car	{E.tipo = 'car' Si TablaCadenas.existe(car) Entonces E.dir= TablaCadena.getIndiceStr(car) Sino E.dir=TablaCadena.append (car) Fin Si
	}
V → id V_C	{Si STS.getCima().existe(id) Entonces ID = id Si V_C.codigo_est=true Entonces V.dir=newTemp() V.tipo =V_C.tipo

```

        gen(V.dir '=' id['V_C.des'])
        V.base = id.dir
        V.codigo_est= true
        V.des = V_C.des
    Sino
        V.dir = id)
        V.tipo= STS.getCima().getTipo(id)
        V.codigo_est= false
    Sino
        error('El id no existe en la pila')
    Fin Si
}

V_C → D_S_T {V_C.tipo =D_S_T.tipo
               V_C.des =D_S_T.des
               V_C.codigo_est =D_S_T.codigo_est}

V_C → A {V_C.tipo =A.tipo
          V_C.des = A.dir
          V_C.codigo_est = true }

V_C → (P){ Si STS.getGlobal().getVar(ID)= 'func' Entonces
            lista = STS.getGlobal().getListaArgs(ID)
            num = STS.getGlobal().getNumArgs(ID)
            Si num = P.num Entonces
                Para cada i = 0 hasta i = num hacer
                    Si lista[i] !=P.lista[i] entonces
                        Error("Los parámetros de la función no coinciden
con la función")
                    Fin si}
            Fin si}

D_S_T → D_S_T1.id {Si D_S_T1.estructura = true Entonces
                   Si D_S_T1.tabla.existe(id) Entonces
                       D_S_T.des = D_S_T1.des
+D_S_T.tabla1.getDir(id)
                       tipoTemp=D_S_T1.tabla.getTipo(id)
                       estTemp = D_S_T1.tabla
                       TT.getName(tipoTemp)
                       Si estTemp = 'struct' Entonces
                           D_S_T.estructura= true
                           D_S_T.tabla=D_S_T1.tabla
                           .TT.getTipoBase(tipoTemp).tabla
                       Sino
                           D_S_T.estructura= false
                           D_S_T.tabla= NULO
                           D_S_T.tipo =D_S_T1.tabla.getTipo(id)
                       FinSi
                       D_S_T.codigo_est=true
                   Sino
                       error("El id no ha sido declarado")
                   FinSi

```

```

                Sino
                    error("La estructura no ha sido declarada")
                FinSi
            }

```

```

D_S_T → E {tipoTemp = STS.getCima().getTipo(id)
    Si STT.getCima().getNombre(tipoTemp) ='struct' Entonces
        D_S_T.estructura= true
        D_S_T.tabla= STT.getCima().getTipoBase(tipoTemp).tabla
        D_S_T.des = 0
    Sino
        D_S_T.estructura= false
        D_S_T.tipo = STT.getCima().getTipo(id)
    Fin Si
    D_S_T.codigo_est=false}

```

```

A → id [E]{A.tipo = STS.getCima().getTipo(ID)
    Si STT.getCima().getNombre(A.tipo) = 'array' Entonces
        Si E.tipo = ent Entonces
            tipoTemp = STT.getCima().getTipoBase(A.tipo)
            tam = STT.getCima().getTam(tipoTemp)
            A.dir = nuevaTemp()
            gen(A.dir='E.dir '*' tam)
        Sino
            error("La expresión no es de tipo entero")
        Fin Si
    Sino
        error("No existe el arreglo")
    Fin Si
}

```

```

A → A1 [E]{A.tipo = STS.getCima().getTipo(A1.tipo)
    Si STT.getCima().getNombre(A.tipo) = 'array' Entonces
        Si E.tipo = ent Entonces
            tipoTemp = STT.getCima().getTipoBase(A.tipo)
            tam = STT.getCima().getTam(tipoTemp)
            dirTemp = nuevaTemp()
            A.dir = nuevaTemp()
            gen(dirTemp '=' E.dir '*' tam)
            gen(A.dir='A1.dir '+' dirTemp)
        Sino
            error("La expresión no es de tipo entero")
        Fin Si
    Sino
        Error("No existe el arreglo")
    Fin Si
}

```

```

P → L_P {P.lista =L_P.lista
    P.num = L_P.num}

```

```

P → {parametros.lista = NULO

```

---

```
    parametros.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}
```

---