Esquema de traducción

---

programa → { STS.push(newTS())

STT.push(newTT())

dir = 0} declaraciones {programa.codigo = funciones.codigo} funciones

---

declaraciones → tipo {typeGBL = tipo.type} lista_var; declaraciones

declaraciones → tipo_registro {typeGBL= tipo registro.type} lista_var; declaraciones

declaraciones → ε

---

tipo_registro → **estructura inicio** declaraciones **fin**

{STS.push(newTS())

STT.push(newTT())

SDir.push(dir)

dir = 0

SymTab = STS.pop()

SymTab.typeTab = STT.pop()

tam = getTam(SymTab)

dir = SDir.pop()

tipo_registro.type =

STT.getTop().insert(”struct”, tam, SymTab)}

---

tipo → base { baseGBL = base.base } tipo_arreglo {tipo.type = tipo arreglo.type}

---

base → **ent** {base.base = STT.getTop().getType(’ent’) }

---

base → **real** {base.base = STT.getTop().getType(’real’)}

---

base → **dreal** {base.base = STT.getTop().getType(’dreal’)}

---

base → **car**{base.base = STT.getTop().getType('car') }

---

base → **sin** {base.base = STT.getTop().getType('sin') }

---

tipo_arreglo → **[num]** tipo_arreglo$_1$

{ **Si** num.type = ent **Entonces**

Si num.dir > 0 **Entonces**

Tipo_arreglo.type =STT.getTop().insert('array',num, tipo_arreglo$_1$.tipo)

Sino

error('...')

Fin Si

Sino

error('...')

Fin Si }

---

tipo_arreglo→ ε {tipo arreglo.type = baseGBL}

---

lista_var → lista_var$_1$**, id**

{

Si STS.getTop().existe(id) **Entonces**

STS.getTop().insert(id, typeGBL, dir, 'var', null, null)

dir ← dir + STT.getTop().getTam(typeGBL)

Sino

error('...')

Fin Si

}

---

lista var → **id**

{**Si** STS.getTop().existe(id) **Entonces**

STS.getTop().insert(id, typeGBL, dir, 'var', null, null)

dir ← dir + STT.getTop().getTam(typeGBL)

Sino

error('...')

Fin Si
}

---

funciones → **def** tipo **id(**argumentos**)**

inicio declaraciones sentencias **fin** funciones

{

Si no STS.getGlobal().existe(id) **Entonces**

STS.push(newTS())

STT.push(newTT())

SDir.push(dir)

dir = 0

listaRET = newListRet()

Si cmpRet(lista_retorno, tipo.type) **Entonces**

L = newLabel()

backpatch(sentencias.nextlist, L)

genCode(label L)

STS.pop()

STT.pop()

Sino

error('...')

Fin Si

Sino

error('...')

Fin Si

}

---

funciones → ε

---

argumentos → lista_arg {argumentos.lista = lista_arg.lista

argumentos.num = lista_arg.num}

---

argumentos → **sin** { argumentos.lista = NULO

argumentos.num = 0 }

---

lista_arg → lista_arg$_1$ {lista_arg.lista = lista_arg$_1$.lista}**,** arg{lista_arg.lista.append(arg.type)

lista_arg.num = lista_arg$_1$.num + 1}

---

lista_arg → {lista_arg.lista = newList()} arg {lista_arg.lista.append(arg.type)

lista_arg.num = 1}

---

arg → tipo_arg **id**

{

Si no STS.getTop().existe(id) **Entonces**

STS.getTop().append(id, tipo.type, dir, 'arg', NULO, NULO)

dir←dir + STT.getTop().getTam(tipo.type)

arg.type = tipo.type

Sino

error(...)

Fin Si

}

---

tipo_arg → base {baseGBL = base.base} param_arr {tipo.type = param arr.type}

---

param_arr → **[ ]** param_arr {Param_arr.type = STT.getTop().insert('array), 0, $param\_arr_1$.tipo)}

---

param arr → ε {param_arr.type = baseGBL }

---

sentencias→$sentencias_1$ sentencia

{L = newLabel()

backpatch($sentencias_1$.nextlist, L)

genCode(label L)}

---

sentencia → **si** e_bool **entonces** $sentencia_1$ **fin**

{

L = newLabel()

backpatch(e_bool.truelist, L)

sentencia.nextlist =combinar(e_bool.falselist, $Sentencia_1$.nextlist)

genCode(label L)

}

---

sentencia →**si** e bool **entonces** $sentencia_1$ **sino** $sentencia_2$ **fin**

{L_1 = newLabel()

L_2 = newLabel()

backpatch(e_bool.truelist, L_1)

backpatch(e_bool.falselist, L_2)

sentencia.nextlist = combinar($sentencia_1$.nextlist, $sentencia_2$.nextlist)

genCode(label L_1)

genCode('goto' $sentencia_1$.nextlist[0])

genCode(label L_2)}

---

sentencia →**mientras** e_bool **hacer** sentencia$_1$ **fin**

{L_1 = newLabel()

L_2 = newLabel()

backpatch(sentencia$_1$.nextlist, L_1)

backpatch(e_bool.truelist, L_2)

sentencia.nextlist = e_bool.falselist

genCode(label L_1)

genCode(label L_2)

genCode('goto' sentencia$_1$.nextlist[0])}

---

sentencia → **hacer** sentencia$_1$ **mientras** e_bool;

{L = newLabel()

genCode("label" L)

batckbatch(sentencia$_1$.nextlist, L)}

---

sentencia → **segun (**variable**) hacer** casos predeterminado **fin**

{L_1 = newLabel()

prueba = combinar(casos.prueba,predeterminado.prueba)

backpatch(casos.nextlist, L_2)

sustituir("??", variable.dir, prueba)}

---

sentencia → variable **:=** expresion **;**

{dir = reducir(expresion.dir, epresion.type, variable.type)

Si variable.code_est = true **Entonces**

genCode(variable.base'['variable.des']' '=' dir)

Sino

genCode(variable.dir '=' dir)

Fin Si}

---

sentencia → **escribir** expresion {gen(”write” expresion.dir)}**;**

---

sentencia → **leer** variable {gen(”read” variable.dir) }**;**

---

sentencia → devolver {genCode(”return”) };

---

sentencia → **devolver** expresion{genCode(”return” expresion.dir)}**;**

sentencia → **terminar;** {index = newIndex()

sentencia.nextlist = newIndexList(index)

genCode(”goto” index)}

---

sentencia → **inicio** sentencias$_1$ {sentencia.nextlist = sentencia$_1$.nextlist} **fin**

---

casos → casos$_1$ **caso num:** sentencia

{casos.nextlist =combinar(casos.nextlist, sentencia$_1$.nextlist)

L = newLabel()

/*Indica el inicio del codigo para la sentencia*/ ´

genCode(”label” L)

casos.prueba = casos$_1$.prueba

casos.prueba.append(if ”??” ”==” num.dir ”goto” L)}

---

casos → **caso num:** sentencia{casos.prueba = newCode()

L = newLabel()

/*Indica el inicio del codigo para la sentencia*/ ´

genCode(”label” L)

casos.prueba.append(if ”??” ”==” num.dir ”goto” L)

casos.nextlist = sentencia.nextlist}

---

predeterminado → **pred:** sentencia {predeterminado.prueba = newCode()

L = newLabel()

/*Indica el inicio del codigo para la sentencia*/ ´

genCode(”label” L)

predeterminado.prueba.append(”goto” L)}

---

predeterminado → ε {pretederminado.prueba = NULO }

---

e_bool → e_bool$_1$ **o** e_bool$_2$

{L = newLabel()

backpatch(e_bool$_1$.falselist, L)

e_bool.truelist = combinar(e_bool$_1$.truelist,

e_bool$_2$.truelist)

e_bool.falselist = e_bool$_2$.falselist

genCode(label L)}

---

e_bool → e_bool **y** e_bool

{L = newLabel()

backpatch(e bool$_1$.truelist, L)

e_bool.truelist = e_bool$_1$.truelist

e_bool.falselist = combinar(e_bool$_1$.falselist,e_bool$_2$.falselist)

genCode(label L)}

---

e _bool → **no** e_bool$_1${e_bool.truelist = e_bool$_1$.falselist

e_bool.falselist = e_bool.truelist}

---

e_bool → relacional_op{e_bool.truelist = relacional_op.truelist

e_bool.falselist = relacional_op.falselist}

---

e_bool → **verdadero**{$index_0$ = newIndex()

e_bool.truelist = newIndexList($index_0$)

genCode('goto' $index_0$)}

---

e_bool → **falso**{$index_0$ = newIndex()

e_bool.falselist = newIndexList($index_0$)

genCode('goto' $index_0$)}

---

relacional_op → relacional$_1$ **>** relacional$_2${$index_0$ = newIndex()

$index_1$ = newIndex()

relacional.truelist = newIndexList($index_0$)

relacional.falselist = newIndexList($index_1$)

genCode('if' relacional$_1$.dir > relacional$_2$ 'goto' $index_0$)

genCode('goto' $index_1$)}

---

relacional_op → relacional **<** relacional{$index_0$ = newIndex()

$index_1$ = newIndex()

relacional.truelist = newIndexList($index_0$)

relacional.falselist = newIndexList($index_1$)

genCode('if' relacional$_1$.dir < relacional$_2$ 'goto' $index_0$)

genCode('goto' $index_1$)}

---

relacional_op → relacional **<=** relacional{$index_0$ = newIndex()

$index_1$ = newIndex()

relacional.truelist = newIndexList($index_0$)

relacional.falselist = newIndexList($index_1$)

genCode('if' relacional$_1$.dir <= relacional$_2$ 'goto' $index_0$)

genCode('goto' $index_1$)}

---

relacional_op → relacional **>=** relacional {$index_0$ = newIndex()

$index_1$ = newIndex()

relacional.truelist = newIndexList($index_0$)

relacional.falselist = newIndexList($index_1$)

genCode('if' $relacional_1$.dir >= $relacional_2$ 'goto' $index_0$)

genCode('goto' $index_1$)}

---

relacional_op → relacional **<>** relacional {$index_0$ = newIndex()

$index_1$ = newIndex()

relacional.truelist = newIndexList($index_0$)

relacional.falselist = newIndexList($index_1$)

genCode('if' $relacional_1$.dir > $relacional_2$ 'goto' $index_0$)

genCode('goto' $index_1$)}

---

relacional_op → relacional **<>** relacional {$index_0$ = newIndex()

$index_1$ = newIndex()

relacional.truelist = newIndexList($index_0$)

relacional.falselist = newIndexList($index_1$)

genCode('if' $relacional_1$.dir > $relacional_2$ 'goto' $index_0$)

genCode('goto' $index_1$)}

---

relacional → expresion {relacional.dir = expresion.dir

relacional.tipo = expresion.tipo}

---

expresion → expresion$_1$ **+** expresion$_2$ {expresion.type = max(expresion$_1$.type, expresion$_2$.type)

expresion.dir = newTemp()

dir$_1$ = ampliar(expresion$_1$.dir, epxresion$_1$.type,

expresion.type)

dir$_2$ = ampliar(expresion$_2$.dir, epxresion$_2$.type,

expresion.type)

getCode(expresion.dir '=' dir$_1$ '+' dir$_2$) }

---

expresion → expresion$_1$ **-** expresion$_2$ {expresion.type = max(expresion$_1$.type, expresion$_2$.type)

expresion.dir = newTemp()

dir$_1$ = ampliar(expresion$_1$.dir, epxresion$_1$.type,

expresion.type)

dir$_2$ = ampliar(expresion$_2$.dir, epxresion$_2$.type,

expresion.type)

getCode(expresion.dir '=' dir$_1$ '-' dir$_2$) }

---

expresion → expresion$_1$ ***** expresion$_2$ {expresion.type = max(expresion$_1$.type, expresion$_2$.type)

expresion.dir = newTemp()

dir$_1$ = ampliar(expresion$_1$.dir, epxresion$_1$.type,

expresion.type)

dir$_2$ = ampliar(expresion$_2$.dir, epxresion$_2$.type,

expresion.type)

getCode(expresion.dir '=' dir$_1$ '*' dir$_2$) }

---

expresion → expresion$_1$ **/** expresion$_2$ {expresion.type = max(expresion$_1$.type, expresion$_2$.type)

expresion.dir = newTemp()

dir$_1$ = ampliar(expresion$_1$.dir, epxresion$_1$.type,

expresion.type)

dir_2 = ampliar(expresion$_2$.dir, epxresion$_2$.type,

expresion.type)

getCode(expresion.dir '=' dir_1 '/' dir_2) }

---

expresion → expresion$_1$ **%** expresion$_2$

{**Si** expresion$_1$.type = entero **and** expresion$_2$.type = entero **Entonces**

expresion.type = max(expresion$_1$.type,expresion$_2$.type)

expresion.dir = newTemp()

dir_1 = ampliar(expresion$_1$.dir, epxresion$_1$.type,expresion.type)

dir_2 = ampliar(expresion$_2$.dir, epxresion$_2$.type,expresion.type)

getCode(expresion.dir '=' dir_1 '+' dir_2)

Sino

error('...')

Fin Si}

---

expresion →**(**expresion$_1$ **)** {epxresion.type = expresion$_1$.type

expresion.dir = expresion$_1$.dir}

---

expresion → variable{epxresion.type = variable.type

expresion.dir = variable.dir}

---

expresion →**num**{epxresion.type = num.type

expresion.dir = num.dir}

---

expresion →**cadena**

{expresion.type ='string'

Si TablaCadenas.existe(cadena) **Entonces**

expresion.dir= TablaCadena.getIndexStr(cadena)

Sino

expresion.dir=TalbaCadena.insert(cadena)

Fin Si}

---

expresion →**caracter**{expresion.type =’car’

Si TablaCadenas.existe(car) **Entonces**

expresion.dir= TablaCadena.getIndexStr(car)

Sino

expresion.dir=TalbaCadena.insert(car)

Fin Si}

---

variable → **id** variable_comp

{**Si** STS.getTop().existe(id) **Entonces**

IDGBL = id

Si variable_com.code_est=true **Entonces**

variable.dir=newTemp()

variable.type = variable_com.type

genCode(variable.dir ’=’ id’[’ variable com.des’]’)

variable.base = id.dir

variable.code_est= true

variable.des = variable com.des

Sino

variable.dir = id)

variable.type = STS.getTop().getType(id)

variable.code_est= false

Sino

error(’...’)

Fin Si}

---

variable_comp → dato_est_sim{Variable_comp.type = dato_est_sim.type

Variable_comp.des = dato_est_sim.des

Variable_comp.code_est = dato_est_sim.code_est}

---

variable_comp → arreglo {variable_comp.type = arreglo.type

variable_comp.des = arreglo.dir

variable_comp.code_est = true}

---

variable_comp → **(** parametros **)** {**Si** STS.getGlobal().getVar(IDGBL)= 'func' **Entonces**

lista = STS.getGlobal().getListaArgs(IDGBL)

num = STS.getGlobal().

Si (Esta incompleto en el documento)}

---

dato_est_sim → dato_est_sim **.id**

{

Si dato_est_sim$_1$.estructura = true **Entonces**

Si dato_est_sim$_1$.tabla.existe(id) **Entonces**

dato_est_sim.des = dato_est_sim$_1$.des +dato_est_sim.tabla$_1$.getDir(id)

typeTemp=dato_est_sim$_1$.tabla.getType(id)

estTemp = dato_est_sim$_1$.tabla.tablaTipos.getName(typeTemp)

Si estTemp = 'struct' **Entonces**

dato_est_sim.estructura= true

dato_est_sim.tabla= dato_est_sim.tabla

.tablaTipos.getTipoBase(typeTemp).tabla

Sino

dato_est_sim.estructura= false

dato_est_sim.tabla= NULO

dato_est_sim.type = dato est sim1.tabla.getType(id)

FinSi

dato_est_sim.code_est=true

Sino

error(...)

FinSi

Sino

error(...)

FinSi

}

---

dato_est_sim → ε

{typeTemp = STS.getTop().getType(id)

Si STT.getTop().getName(typeTemp) =’struct’ **Entonces**

dato_est_sim.estructura= true

dato_est_sim.tabla= STT.getTop().getTipoBase(typeTemp).tabla

dato_est_sim.des = 0

Sino

dato_est_sim.estructura= false

dato_est_sim.type = STT.getTop().getType(id)

Fin Si

dato_est_sim.code est=false}

---

arreglo → **[** expresion **]**

{arreglo.type = STS.getTop().getType(IDGBL)

Si STT.getTop().getName(arreglo.type) = ’array’ **Entonces**

Si expresion.type = entero **Entonces**

typeTemp = STT.getTop().getTypeBase(arreglo.type)

tam = STT.getTop().getTam(typeTemp)

arreglo.dir = newTemp()

genCode(arreglo.dir'=' expresion.dir '*' tam)

Sino

error(...)

Fin Si

Sino

error(...)

Fin Si}

---

arreglo → arreglo$_1$ {arreglo.type = STS.getTop().getType(arreglo$_1$.type)} **[** expresion **]**

{**Si** STT.getTop().getName(arreglo.type) = 'array' **Entonces**

Si expresion.type = entero **Entonces**

typeTemp = STT.getTop().getTypeBase(arreglo.type)

tam = STT.getTop().getTam(typeTemp)

dirTemp = newTemp()

arreglo.dir = newTemp()

genCode(dirTemp'=' expresion.dir '*' tam)

genCode(arreglo.dir'=' arreglo$_1$.dir '+' dirTemp)

Sino

error(...)

Fin Si

Sino

error(...)

Fin Si}

---

parametros → lista_param {parametros.lista = lista_param.lista

parametros.num = lista_param.num}

---

parametros→ ε {parametros.lista = NULO

parametros.num = 0}

---

lista_param → lista_param$_1$ {lista_param.lista = lista_param$_1$.lista},

expresion{lista_param .lista.append(expresion.type)

lista_param.num = lista_param$_1$ + 1}

---

lista_param → expresion{lista_param.lista = newList()

lista_param.lista.append(expresion.type)

lista_param.num = 1}
