CUARTA FASE: FINALIZACIÓN DEL PROCESADOR PARA Tiny(1)

Javier Gómez Moraleda
Mario Quiñones Pérez
Grupo 18

1. Especificación del procesamiento de vinculación.

global ts //tabla de símbolos

vincula(Programa(Decs,Is)) =
    if(hayDeclaraciones())
        vinculacion(Decs)
        vinculacionPointer(Decs)
    vincula(Is)

vinculacion(Decs_una(Dec)) =
    vinculacion(Dec)

vinculacion(Decs_muchas(Decs,Dec)) =
    vinculacion(Decs)
    vinculacion(Dec)

vinculacion(DecVar(T,id)) =
    vinculacion(T)
    recolecta(id,$)

vinculacion(DecTipo(T,id)) =
    vinculacion(T)
    recolecta(id,$)

vinculacion(DecProc(id,Ps,B)) =
    recolecta(id,$)
    vinculacion(Ps)
    vinculacion(B)

vinculacion(ref(id)) =
    if (existe_id(id,ts) entonces)
        $.vinculo = valorDe(ts,id)
    else
        error

vinculacion(Bloque_inst(B)) =
    anida()
    vinculacion(B)
    desanida()

vinculacion(Call(id,exps)) =
    if (!idDuplicadoTodos(id))
        printError(id, tError.IdNoDeclarado);
    else
        setVinculo(getDec(id))

vinculacion(exps)

vinculacion(Delete (exp)) =
vinculacion(exp)

vinculacion(New_cons (exp)) =
vinculacion(exp)

vinculacion(Nl()) =
skip

vinculacion(Write (exp)) =
vinculacion(exp)

vinculacion(Read (exp)) =
vinculacion(exp)

vinculacion(While_inst (exp, instrucciones)) =
vinculacion(exp)
vinculacion(instrucciones)

vinculacion(If_else (exp, instrucciones, instrucciones_else)) =
vinculacion(exp)
vinculacion(instrucciones)
vinculacion(instrucciones_else)

vinculacion(If_inst (exp, instrucciones)) =
vinculacion(exp)
vinculacion(instrucciones)

vinculacion(Asig (exp0, exp1)) =
vinculacion(exp0)
vinculacion(exp1)

vinculacion(Insts_muchas (instrucciones, instruccion)) =
vinculacion(instrucciones)
vinculacion(instruccion)

vinculacion(Insts_una (instruccion)) =
vinculacion(instruccion)

vinculacion(Lista_inst_empty()) =
skip

vinculacion(Lista_inst_una (instruccion)) =
vinculacion(instruccion)

vinculacion(Lista_inst_muchas (instrucciones, instruccion)) =
    vinculacion(instrucciones)
    vinculacion(instruccion)

vinculacion(ParamForm (id, tipo, paramForm)) =
    vinculacion(tipo)
    recolectaAct(id, paramForm)

vinculacion(Pformal_ref (id, tipo, paramForm)) =
    vinculacion(tipo)
    recolectaAct(id, paramForm)

vinculacion(ParamForms_muchos (params, param)) =
    vinculacion(params)
    vinculacion(param)

vinculacion(ParamForms_uno (param)) =
    vinculacion(param)

vinculacion(ParamForms_empty()) =
    skip

vinculacion(Campos_muchos (campos, campo)) =
    vinculacion(campos)
    vinculacion(campo)

vinculacion(Campo_uno (campo)) =
    vinculacion(campo)

vinculacion(Camp (tipo)) =
    vinculacion(tipo)

vinculacion(Bloque_prog (programa)) =
    vinculacion(programa)

vinculacion(No_bloque()) =
    skip

vinculacion(Exp_muchas (expresiones, expresion)) =
    vinculacion(expresiones)
    vinculacion(expresion)

vinculacion(Exp_una (expresion)) =
    vinculacion(expresion)

vinculacion(Suma (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Resta (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(And (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Or (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Menor (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Mayor (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(MenorIgual (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(MayorIgual (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Igual (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Distinto (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Mul (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Div (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(Percent (arg0, arg1)) =
    vinculacion(arg0)
    vinculacion(arg1)

vinculacion(MenosUnario (arg0)) =
	vinculacion(arg0)

vinculacion(Not (arg0)) =
	vinculacion(arg0)

vinculacion(Corchete (arg0, arg1)) =
	vinculacion(arg0)
	vinculacion(arg1)

vinculacion(Punto (exp)) =
	vinculacion(exp)

vinculacion(Flecha (exp)) =
	vinculacion(exp)

vinculacion(Star (arg0)) =
	vinculacion(arg0)

vinculacion(Id (id)) =
	if (!idDuplicadoTodos(id))
		error
	else
		setVinculo(getDec(id));

vinculacion(Tipo_Id (id)) =
	if (!idDuplicadoTodos(id))
		error
	else
		setVinculo(getDec(id));

vinculacion(Array (tipo_array)) =
	vinculacion(tipo_array)

vinculacion(Record (campos)) =
	vinculacion(campos)
	$.tipo =campos.tipo

vinculacion(Pointer (tipo)) =
// Segunda pasada
	skip

vinculacion(Decs_una(Dec)) =
	vinculacion(Dec)

vinculacion(Decs_muchas(Decs,Dec)) =
	vinculacion(Decs)
	vinculacion(Dec)

vinculacion(int_cons())=
skip

vinculacion(real_cons())=
skip

vinculacion(bool_cons())=
skip

vinculacion(string_cons())=
skip

// Segunda pasada sobre declaraciones: se vinculan los ref en ‘pointer’
vinculacionPointer(Decs_una(Dec)) =
vinculacionPointer(Dec)

vinculacionPointer(Decs_muchas(Decs,Dec)) =
vinculacionPointer(Decs)
vinculacionPointer(Dec)

vinculacionPointer(DecVar(T,id)) =
vinculacionPointer(T)

vinculacionPointer(DecTipo(T,id)) =
vinculacionPointer(T)

vinculacionPointer(DecProc(id,Ps,B)) =
skip

vinculacionPointer(ref(id)) =
skip

vinculacionPointer(Int())=
skip

vinculacionPointer(real_cons())=
skip

vinculacionPointer(bool_cons())=
skip

vinculacionPointer(string_cons())=
skip

vinculacionPointer(pointer(T)) =
si idDuplicadoTodos(id) entonces
T.vinculo = getDec(id)
si no error

2. Especificación del procesamiento de comprobación de tipos.

chequeo_tipo(Programa(Decs,Is)) =
    if(hayDeclaraciones())
        chequeo_tipo(declaraciones)
    chequeo_tipo(instrucciones)
    $.tipo = ambos_ok(Ds.tipo,Is.tipo)

chequeo_tipo(Decs_muchas(Dec)) =
    chequeo_tipo(declaraciones)
    chequeo_tipo(declaracion)

chequeo_tipo(Decs_una(Decs,Dec)) =
    chequeo_tipo(declaracion)

chequeo_tipo(DecProc(id,Ps,B)) =
    chequeo_tipo(Ps)
    chequeo_tipo(B)

chequeo_tipo(DecTipo(T,id)) =
    chequeo_tipo(val)

chequeo_tipo(DecVar(T,id)) =
    chequeo_tipo(val)
    $.tipo = val.tipo

chequeo_tipo(Bloque_inst(B)) =
    chequeo_tipo(bloque)
    $.tipo = bloque.tipo

chequeo_tipo(Call(id,exps)) =
    chequeo_tipo(exps)

    if (getVinculo().tipo == ok())
        $.tipo = ok()
     else
        $.tipo = error()

chequeo_tipo(Delete (exp)) =
    chequeo_tipo(exp)

    if (delete.exp().tipo == tNodo.POINTER)
        $.tipo = ok()
     else
        $.tipo = error()

chequeo_tipo(New_cons (exp)) =

```
chequeo_tipo(exp)

if (new_cons.exp().tipo == tNodo.POINTER)
        $.tipo = ok()
 else
        $.tipo = error()
```

chequeo_tipo(Nl()) =

```
$.tipo = ok()
```

chequeo_tipo(Write(exp)) =

```
chequeo_tipo(exp)
if (extipo == tNodo.LIT_ENT || extipo == tNodo.LIT_REAL || extipo ==
tNodo.STRING || extipo == tNodo.BOOL)
        $.tipo = ok()
 else
        $.tipo = error()
```

chequeo_tipo(Read(exp)) =

```
chequeo_tipo(exp)
if (exesDesignador() && (extipo == tNodo.LIT_ENT || extipo == tNodo.LIT_REAL ||
extipo == tNodo.STRING))
        $.tipo = ok()
 else
        $.tipo = error()
```

chequeo_tipo(While_inst(exp, instrucciones)) =

```
chequeo_tipo(exp)
chequeo_tipo(instrucciones)

if (extipo == tNodo.BOOL && instrucciones.tipo == ok())
        $.tipo = ok()
 else
        $.tipo = error()
```

chequeo_tipo(If_else(exp, instrucciones, instrucciones_else)) =

```
chequeo_tipo(exp)
chequeo_tipo(instrucciones)
chequeo_tipo(instrucciones_else)
if (extipo == tNodo.BOOL && instrucciones.tipo == ok() && instrucciones_else.tipo
== ok())
        $.tipo = ok()
 else
        $.tipo = error()
```

chequeo_tipo(If_inst(exp, instrucciones)) =

```
        chequeo_tipo(exp)
        chequeo_tipo(instrucciones)
        if (exptipo == tNodo.BOOL && instrucciones.tipo == ok())
                $.tipo = ok()
         else
                $.tipo = error()

chequeo_tipo(Asig  (exp0, exp1)) =
        chequeo_tipo(exp0)
        chequeo_tipo(exp1)
        if (exp0esDesignador())
                if (compatible(exp0, exp1))
                        $.tipo = ok()
                else
                        $.tipo = error()
         else
                $.tipo = error()

chequeo_tipo(Insts_muchas (instrucciones, instruccion)) =
        chequeo_tipo(instrucciones)
        chequeo_tipo(instruccion)
        if (instrucciones.tipo == ok() && instruccion.tipo == ok())
                $.tipo = ok()
         else
                $.tipo = error()

chequeo_tipo(Insts_una (instruccion)) =
        chequeo_tipo(instruccion)
        insts.setTipo(insts.instruccion().tipo);

chequeo_tipo(Lista_inst_empty lista_inst_empty()) =
        $.tipo = ok()

chequeo_tipo(Lista_inst_una (instruccion)) =
        chequeo_tipo(instruccion)
        $.tipo = instruccion.tipo

chequeo_tipo(Lista_inst_muchas(instrucciones, instruccion)) =
        chequeo_tipo(instrucciones)
        chequeo_tipo(instruccion)
        if (instrucciones.tipo == ok() && instrucciontipo == ok())
                $.tipo = ok()
         else
                $.tipo = error()

chequeo_tipo(ParamForm (id, tipo, paramForm)) =
        $.tipo = tipo.tipo
```

chequeo_tipo(Pformal_ref (id, tipo, paramForm)) =
	$.tipo = tipo.tipo

chequeo_tipo(ParamForms_muchos (params, param)) =
	chequeo_tipo(params)
	chequeo_tipo(param)
	if (params.tipo == ok()))
		$.tipo = ok()
	 else
		$.tipo = error()

chequeo_tipo(ParamForms_uno (param)) =
	chequeo_tipo(param)
	$.tipo = param.tipo

chequeo_tipo(ParamForms_empty()) =
	$.tipo = ok()

chequeo_tipo(Campos_muchos (campos, campo)) =
	chequeo_tipo(campos)
	chequeo_tipo(campo)
	$.setRecord(campos, campo);

chequeo_tipo(Campo_uno (campo)) =
	chequeo_tipo(campo)
	$.setRecord(campo);

chequeo_tipo(Camp (tipo)) =
	chequeo_tipo(tipo)
	$.tipo = tipo.tipo

chequeo_tipo(Bloque_prog(programa)) =
	chequeo_tipo(programa)
	$.tipo = programa.tipo

chequeo_tipo(No_bloque()) =
	$.tipo = ok()

chequeo_tipo(Lista_exp_empty()) =
	$.tipo = ok()

chequeo_tipo(Exp_muchas (expresiones, expresion)) =
	chequeo_tipo(expresiones)
	chequeo_tipo(expresion)
	if (expresiones.tipo == ok())
		$.tipo = ok()
	 else

```
            $.tipo = error()

chequeo_tipo(Exp_una (expresion)) =
        chequeo_tipo(expresion)
        $.tipo = expresion.tipo

chequeo_tipo(Suma (arg0, arg1)) =
        chequeo_tipo(arg0)
        chequeo_tipo(arg1)

        if (arg0.tipo == tNodo.LIT_ENT && arg1.tipo == tNodo.LIT_ENT)
                $.tipo = tNodo.LIT_ENT
         else if (arg0.tipo == tNodo.LIT_REAL && arg1.tipo == tNodo.LIT_REAL)
                $.tipo = tNodo.LIT_REAL
         else
                $.tipo = error()

chequeo_tipo(Resta (arg0, arg1)) =
        chequeo_tipo(arg0)
        chequeo_tipo(arg1)

        if (arg0.tipo == tNodo.LIT_ENT && arg1.tipo == tNodo.LIT_ENT)
                $.tipo = tNodo.LIT_ENT
         else if (arg0.tipo == tNodo.LIT_REAL && arg1.tipo == tNodo.LIT_REAL)
                $.tipo = tNodo.LIT_REAL
         else
                $.tipo = error()

chequeo_tipo(And (arg0, arg1)) =
        chequeo_tipo(arg0)
        chequeo_tipo(arg1)

        if (arg0tipo == tNodo.BOOL && arg1.tipo == tNodo.BOOL)
                $.tipo = tNodo.BOOL
         else
                $.tipo = error()

chequeo_tipo(Or (arg0, arg1)) =
        chequeo_tipo(arg0)
        chequeo_tipo(arg1)

        if (arg0tipo == tNodo.BOOL && arg1.tipo == tNodo.BOOL)
                $.tipo = tNodo.BOOL
         else
                $.tipo = error()
```

chequeo_tipo(Menor (arg0, arg1)) =
	chequeo_tipo(arg0)
	chequeo_tipo(arg1)

	if (compatible(arg0, arg1))
		$.tipo = tNodo.BOOL
	 else
		$.tipo = error()

chequeo_tipo(Mayor (arg0, arg1)) =
	chequeo_tipo(arg0)
	chequeo_tipo(arg1)

	if (compatible(arg0, arg1))
		$.tipo = tNodo.BOOL
	 else
		$.tipo = error()

chequeo_tipo(MenorIgual (arg0, arg1)) =
	chequeo_tipo(arg0)
	chequeo_tipo(arg1)

	if (compatible(arg0, arg1))
		$.tipo = tNodo.BOOL
	 else
		$.tipo = error()

chequeo_tipo(MayorIgual (arg0, arg1)) =
	chequeo_tipo(arg0)
	chequeo_tipo(arg1)

	if (compatible(arg0, arg1))
		$.tipo = tNodo.BOOL
	 else
		$.tipo = error()

chequeo_tipo(Igual (arg0, arg1)) =
	chequeo_tipo(arg0)
	chequeo_tipo(arg1)

	if (compatible(arg0, arg1))
		$.tipo = tNodo.BOOL
	 else
		$.tipo = error()

chequeo_tipo(Distinto (arg0, arg1)) =
	chequeo_tipo(arg0)
	chequeo_tipo(arg1)

```
        if (compatible(arg0, arg1))
                $.tipo = tNodo.BOOL
         else
                $.tipo = error()

chequeo_tipo(Mul (arg0, arg1)) =
        chequeo_tipo(arg0)
        chequeo_tipo(arg1)

        if (arg0.tipo == tNodo.LIT_ENT && arg1.tipo == tNodo.LIT_ENT)
                $.tipo = tNodo.LIT_ENT
         else if (arg0.tipo == tNodo.LIT_REAL && arg1.tipo == tNodo.LIT_REAL)
                $.tipo = tNodo.LIT_REAL
         else
                $.tipo = error()

chequeo_tipo(Div (arg0, arg1)) =
        chequeo_tipo(arg0)
        chequeo_tipo(arg1)
        if (arg0.tipo == tNodo.LIT_ENT && arg1.tipo == tNodo.LIT_ENT)
                $.tipo = tNodo.LIT_ENT
         else if (arg0.tipo == tNodo.LIT_REAL && arg1.tipo == tNodo.LIT_REAL)
                $.tipo = tNodo.LIT_REAL
         else
                $.tipo = error()

chequeo_tipo(Percent (arg0, arg1)) =
        chequeo_tipo(arg0)
        chequeo_tipo(arg1)
        if (arg0.tipo == tNodo.LIT_ENT && arg1.tipo == tNodo.LIT_ENT)
                $.tipo = tNodo.LIT_ENT
         else if (arg0.tipo == tNodo.LIT_REAL && arg1.tipo == tNodo.LIT_REAL)
                $.tipo = tNodo.LIT_REAL
         else
                $.tipo = error()

chequeo_tipo(MenosUnario (arg0)) =
        chequeo_tipo(arg0)

        if (arg0.tipo == tNodo.LIT_ENT)
                $.tipo = tNodo.LIT_ENT
         else if (arg0.tipo == tNodo.LIT_REAL))
                $.tipo = tNodo.LIT_REAL
         else
                $.tipo = error()
```

chequeo_tipo(Not (arg0)) =
	chequeo_tipo(arg0)

	if (arg0.tipo == tNodo.BOOL)
		$.tipo = tNodo.BOOL
	 else
		$.tipo = error()

chequeo_tipo(Corchete (arg0, arg1)) =
	chequeo_tipo(arg0)
	chequeo_tipo(arg1)

	if (arg1.tipo == tNodo.LIT_ENT && arg0.tipo == tNodo.ARRAY)
		$.tipo = campos.get(id).tipo
	 else
		$.tipo = error()

chequeo_tipo(Punto (exp)) =
	chequeo_tipo(exp)

	if (ref(exp).tipo == tNodo.RECORD) =
		$.tipo = ref(exp)
	 else
		$.tipo = error()

chequeo_tipo(Flecha (exp)) =
	chequeo_tipo(exp)

	if (ref(exp).tipo == tNodo.RECORD) =
		$.tipo = ref(exp)
	 else
		$.tipo = error()

chequeo_tipo(Star (arg0)) =
	chequeo_tipo(arg0)

	if (ref(arg0).tipo == tNodo.POINTER)
		$.tipo = ref(arg0).tipo
	 else
		$.tipo = error()

chequeo_tipo(True()) =
	$.tipo = tNodo.BOOL

chequeo_tipo(False()) =
	$.tipo = tNodo.BOOL

chequeo_tipo(LitReal()) =
	$.tipo = tNodo.LIT_REAL

chequeo_tipo(Id(id)) =
	if ($.getVinculo() = DecTipo(id))
		$.tipo = ok()
	else
		$.tipo = error()

chequeo_tipo(LitEnt()) =
	$.tipo = tNodo.LIT_ENT

chequeo_tipo(LitNull()) =
	$.tipo = tNodo.NULL

chequeo_tipo(LitCad()) =
	$.tipo = tNodo.STRING

chequeo_tipo(Bool()) =
	$.tipo = tNodo.BOOL

chequeo_tipo(Int()) =
	$.tipo = tNodo.LIT_ENT

chequeo_tipo(Real()) =
	$.tipo = tNodo.LIT_REAL

chequeo_tipo(String_cons()) =
	$.tipo = tNodo.STRING

chequeo_tipo(Tipo_Id(id)) =
	$.tipo = $.getvinculo().tipo

chequeo_tipo(Array(tipo_array)) =
	chequeo_tipo(tipo_array)
	$.tipo = tipo_array.tipo

chequeo_tipo(Record(campos)) =
	chequeo_tipo(campos)
	$.tipo = campos.tipo

chequeo_tipo(Pointer(tipo)) =
	chequeo_tipo(tipo)
	$.tipo = tipo.tipo

ambos_ok(to,t1) =
	si to=ok() && t1=ok()

```
        return ok()
else
        return error()
```

3. Especificación del procesamiento de asignación de espacio.

```
global dir=0 // contador de direcciones
gobal nivel=0 // nivel de anidamiento

asignacion_espacio(prog(Decs, Is)) =
        if(hayDeclaraciones())
                asignación_espacio(Decs)
        asignación_espacio(Is)

asignación_espacio(decs_muchas(Decs, Dec)) =
        asignación_espacio(Decs)
        asignación_espacio(Dec)

asignación_espacio(decs_una(Dec)) =
        asignación_espacio(Dec)

asignación_espacio(dec_proc(id, Ps, Ds, Is)) =
        ant_dir = dir // se salva el valor de dir
        nivel = nivel + 1 // el nivel se incrementa en 1
        $.nivel = nivel
        dir = 0 // Las direcciones en los procedimientos son relativas al comienzo del
                // registro de activación
        asignación_espacio(Ps)
        asignación_espacio(Ds)
        $.tam_datos = dir // el valor de dir en este punto indica el tamaño de los datos
                          // locales Una vez finalizado el procesamiento del procedimiento,
                             // se restaura los valores de dir y nivel
        dir = ant_dir // se restaura el valor de dir
        nivel = nivel - 1 // se decrementa el nivel en 1

asignación_espacio(dec_tipo(T ,id)) =
        asignación_espacio_tipo(T)
        $.tam_datos = T.tam_datos

asignación_espacio(dec_var(T,id)) =
        $.dir = dir
        $.nivel = nivel
        asignación_espacio_tipo(T)
        $tam_datos = T.tam_datos
        dir = dir + T.tam // se incrementa la dirección en el tamaño requerido por T

asignación_espacio(Bloque_inst(B)) =
        asignación_espacio(B)
```

asignación_espacio(Call(exps)) =
    asignación_espacio(exps)

asignación_espacio(Delete (exp)) =
    asignación_espacio(exp)

asignación_espacio(New_cons (exp)) =
    asignación_espacio(exp)

asignación_espacio(Nl()) =
    skip

asignación_espacio(Write(exp)) =
    asignación_espacio(exp)

asignación_espacio(Read(exp)) =
    asignación_espacio(exp)

asignación_espacio(While_inst(exp, instrucciones)) =
    asignación_espacio(exp)
    asignación_espacio(instrucciones)

asignación_espacio(If_else(exp, instrucciones, instrucciones_else)) =
    asignación_espacio(exp)
    asignación_espacio(instrucciones)
    asignación_espacio(instrucciones_else)

asignación_espacio(If_inst(exp, instrucciones)) =
    asignación_espacio(exp)
    asignación_espacio(instrucciones)

asignación_espacio(Asig (exp0, exp1)) =
    asignación_espacio(exp0)
    asignación_espacio(exp1)

asignación_espacio(Insts_muchas (instrucciones, instruccion)) =
    asignación_espacio(instrucciones)
    asignación_espacio(instruccion)

asignación_espacio(Insts_una (instruccion)) =
    asignación_espacio(instruccion)

asignación_espacio(Lista_inst_empty lista_inst_empty()) =
    skip

asignación_espacio(Lista_inst_una (instruccion)) =
    asignación_espacio(instruccion)

asignación_espacio(Lista_inst_muchas(instrucciones, instruccion)) =
    asignación_espacio(instrucciones)
    asignación_espacio(instruccion)

asignación_espacio(ParamForm (id, tipo, paramForm)) =
    asignacion_espacio(tipo)
    $dir = dir
    $nivel = nivel
    $tam_datos = tipo.tam_datos
    dir = dir + paramForm.tam_datos

asignación_espacio(Pformal_ref (id, tipo, paramForm)) =
    asignacion_espacio(tipo)
    $dir = dir
    $nivel = nivel
    $tam_datos = tipo.tam_datos
    dir = dir + 1

asignación_espacio(ParamForms_muchos (params, param)) =
    asignación_espacio(params)
    asignación_espacio(param)

asignación_espacio(ParamForms_uno (param)) =
    asignación_espacio(param)

asignación_espacio(ParamForms_empty()) =
    skip

asignación_espacio(Campos_muchos (campos, campo)) =
    asignación_espacio(campos)
    asignación_espacio(campo)
    campo.desplazamiento = campos.tam_datos
    $.tam_datos = campos.tam_datos + campo.tam_datos

asignación_espacio(Campo_uno (campo)) =
    asignación_espacio(campo)
    $.tam_datos = campo.tam_datos
    campo.desplazamiento = 0

asignación_espacio(Camp (tipo)) =
    asignación_espacio(tipo)
    $.tam_datos = tipo.tam_datos
    $.basesize = tipo.tam_basico

asignación_espacio(Bloque_prog(programa)) =
    asignación_espacio(programa)

asignación_espacio(No_bloque()) =

skip

asignación_espacio(Lista_exp_empty()) =
skip

asignación_espacio(Exp_muchas (expresiones, expresion)) =
asignación_espacio(expresiones)
asignación_espacio(expresion)

asignación_espacio(Exp_una (expresion)) =
asignación_espacio(expresion)

asignación_espacio(Suma (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(Resta (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(And (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(Or (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(Menor (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(Mayor (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(MenorIgual (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(MayorIgual (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(Igual (arg0, arg1)) =
asignación_espacio(arg0)
asignación_espacio(arg1)

asignación_espacio(Distinto (arg0, arg1)) =
    asignación_espacio(arg0)
    asignación_espacio(arg1)

asignación_espacio(Mul (arg0, arg1)) =
    asignación_espacio(arg0)
    asignación_espacio(arg1)

asignación_espacio(Div (arg0, arg1)) =
    asignación_espacio(arg0)
    asignación_espacio(arg1)

asignación_espacio(Percent (arg0, arg1)) =
    asignación_espacio(arg0)
    asignación_espacio(arg1)

asignación_espacio(MenosUnario (arg0)) =
    asignación_espacio(arg0)

asignación_espacio(Not (arg0)) =
    asignación_espacio(arg0)

asignación_espacio(Corchete (arg0, arg1)) =
    asignación_espacio(arg0)
    asignación_espacio(arg1)
    $.tam_datos = arg0.tam_basico

asignación_espacio(Punto (exp)) =
    asignación_espacio(exp)
    $.tam_datos = exgetVinculo().tam_datos;
    $.tam_basico = exgetVinculo().tam_basico;

asignación_espacio(Flecha (exp)) =
    asignación_espacio(exp)
    $.tam_datos = exgetVinculo().tam_datos;
    $.tam_basico = exgetVinculo().tam_basico;

asignación_espacio(Star (arg0)) =
    asignación_espacio(arg0)
    $.tam_datos = 1
    $.tam_basico = arg0.tam_datos

asignación_espacio(True()) =
    skip

asignación_espacio(False()) =
    skip

asignación_espacio(LitReal()) =
    skip

asignación_espacio(Id(id)) =
    $.direccion = $.vinculo.direccion
    $.nivel = $.vinculo.nivel

asignación_espacio(LitEnt()) =
    skip

asignación_espacio(LitNull()) =
    skip

asignación_espacio(LitCad()) =
    skip

asignación_espacio(Bool()) =
    $.tam_datos = 1

asignación_espacio(Int()) =
    $.tam_datos = 1

asignación_espacio(Real()) =
    $.tam_datos = 1

asignación_espacio(String_cons()) =
    $.tam_datos = 1

asignación_espacio(Tipo_Id(id)) =
    $.direccion = $.vinculo.direccion
    $.tam_datos = $.vinculo.tam_datos
    $.nivel = $.vinculo.nivel
    $.tam_basico = $.vinculo.tam_basico

asignación_espacio(Array(tipo_array, tam)) =
    asignación_espacio(tipo_array)
    $.tam_basico = tipo_array.tam_datos
    $.tam_datos = $.tam_basico * tam

asignación_espacio(Record(campos)) =
    asignación_espacio(campos)
    $.tam_datos = campos.tam_datos

asignación_espacio(Pointer(tipo)) =
    asignación_espacio(tipo)
    $.tam_datos = 1
    $.tam_basico = tipo.tam_datos

4. Descripción del repertorio de instrucciones de la máquina-p necesario para soportar la traducción de Tiny(1) a código-

Funciones Aritméticas	
suma	Desapila los dos primeros valores de la pila y apila su suma
sumaR	Desapila los dos primeros valores de la pila y apila su suma como valores reales
resta	Desapila los dos primeros valores de la pila y apila su resta
restaR	Desapila los dos primeros valores de la pila y apila su resta como valores reales
mul	Desapila los dos primeros valores de la pila y apila su multiplicación
mulR	Desapila los dos primeros valores de la pila y apila su multiplicación como valores reales
div	Desapila los dos primeros valores de la pila y apila su división
divR	Desapila los dos primeros valores de la pila y apila su división como valores reales
percent	Desapila los dos primeros valores de la pila y apila su módulo
percentR	Desapila los dos primeros valores de la pila y apila su módulo como valores reales
and	Desapila los dos primeros valores de la pila y apila su operación and
or	Desapila los dos primeros valores de la pila y apila su operación or
not	Desapila la cima de la pila y apila su contrario
menos	Desapila la cima de la pila y apila su

	negativo
menosR	Desapila la cima de la pila y apila su negativo de un valor real
menor	Desapila los dos primeros valores de la pila y apila true si el segundo valor es menor que el primero
menorR	Desapila los dos primeros valores de la pila y apila true si el segundo valor es menor que el primero, ambos valores Reales
menorS	Desapila las dos primeras cadenas de la pila y apila true si el segundo valor es menor que el primero
mayor	Desapila los dos primeros valores de la pila y apila true si el segundo valor es mayor que el primero
mayorR	Desapila los dos primeros valores de la pila y apila true si el segundo valor es mayor que el primero, ambos valores Reales
mayorS	Desapila las dos primeras cadenas de la pila y apila true si el segundo valor es mayor que el primero
menorig	Desapila los dos primeros valores de la pila y apila true si el segundo valor es menor o igual que el primero
menorigR	Desapila los dos primeros valores de la pila y apila true si el segundo valor es menor o igual que el primero, ambos valores Reales
menorigS	Desapila las dos primeras cadenas de la pila y apila true si el segundo valor es menor o igual que el primero
mayorig	Desapila los dos primeros valores de la pila y apila true si el segundo valor es mayor o igual que el primero
mayorigR	Desapila los dos primeros valores de la

	pila y apila true si el segundo valor es mayor o igual que el primero, ambos valores Reales
mayorigS	Desapila las dos primeras cadenas de la pila y apila true si el segundo valor es mayor o igual que el primero
ig	Desapila los dos primeros valores de la pila y apila true si el segundo valor es igual que el primero
igR	Desapila los dos primeros valores de la pila y apila true si el segundo valor es igual que el primero, ambos valores Reales
igS	Desapila las dos primeras cadenas de la pila y apila true si el segundo valor es igual que el primero
Escritura/Lectura	
writeInt	Desapila el valor de la cima de la pila y lo escribe por pantalla
writeReal	Desapila el valor de la cima de la pila y lo escribe por pantalla
writeBool	Desapila el valor de la cima de la pila y lo escribe por pantalla
writeString	Desapila el valor de la cima de la pila y lo escribe por pantalla
readInt	Lee el valor entero de la entrada por comandos y lo apila
readReal	Lee el valor real de la entrada por comandos y lo apila
readBool	Lee el valor booleano de la entrada por comandos y lo apila
readString	Lee el string de la entrada por comandos y lo apila
Gestión Memoria Dinamica	

alloc(n)	Reserva un bloque de **n** celdas consecutivas en el heap y apila la dirección de comienzo en la pila de evaluación.
dealloc(n)	Desapila una dirección **d** de la pila de evaluación y libera en el heap el bloque de **n** celdas consecutivas que comienza en **d**
Instrucciones de Salto	
irA(d)	Salta a la dirección d
irF(d)	Salta a la dirección d si el valor que desapila de la pila es false
irV(d)	Salta a la dirección d si el valor que desapila de la pila es true
irind	Desapila un valor de la pila y va a la dirección que este marque
Gestión Movimiento de Datos	
apilaInt(i)	Apila el valor entero i
apilaReal(i)	Apila el valor real i
apilaBool(i)	Apila el valor booleano i
apilaString(i)	Apila la cadena de caracteres i
apilaind	Desapila un valor *dir* y apila el valor que tenga la celda dir de la memoria
desapilaind	Desapila dos valores valor *dir* y *valor* guarda *valor* en la celda de memoria con dirección *dir*
mueve(n)	Desapila dos direcciones y copia el contenido de las n celdas consecutivas que comienzan por la primera dirección desapilada a las n celdas que comienzan por la segunda dirección desapilada
Gestión Funciones	
activa(nivel, tam, direccion)	Reserva espacio en el segmento de pila de registros de activación para ejecutar un

	procedimiento que tiene nivel de anidamiento **nivel** y tamaño de datos locales **tam**. Así mismo, almacena en la zona de control de dicho registro **direccion** como dirección de retorno. También almacena en dicha zona de control el valor del display de nivel **nivel**. Por último, apila en la pila de evaluación la dirección de comienzo de los datos en el registro creado.
apilad(nivel)	Apila en la pila el valor del display de nivel **nivel**
desapilad(nivel)	Desapila una dirección de la pila en el display de nivel **nivel**
desactiva(nivel, tam)	Libera el espacio ocupado por el registro de activación actual, restaurando adecuadamente el estado de la máquina. **nivel** indica el nivel de anidamiento del procedimiento asociado; **tam** el tamaño de los datos locales. De esta forma, la instrucción, primero apila la dirección de retorno, luego restaura el valor del display de nivel **nivel** al antiguo valor guardado en el registro, y por último decrementa el puntero de pila de registros de activación en el tamaño ocupado por el registro
dup	Consulta el valor de la cima de la pila y lo vuelve a apilar
stop	Detiene la maquinaP

5. Especificación del procesamiento de etiquetado.

```
global etq=0
global procs = pila_vacia()

etiqueta(Programa(Decs,Is)) =
        $.etqi = etq;
        if(declaraciones != null()) =
                etiqueta(declaraciones)
        etiqueta(instrucciones)
        $.etqs = etq;

etiqueta(Decs_muchas(Decs,Dec)) =
        etiqueta(declaraciones)
        etiqueta(declaracion)

etiqueta(Decs_una(Dec)) =
        etiqueta(declaracion)

etiqueta(DecProc(id,Ps,B)) =
        $.etqi = etq;
        etiqueta(B)
        $.etqs = etq;

etiqueta(Bloque_inst(Bloque_inst(B)) =
        etiqueta(bloque)

etiqueta(Call(id,exps)) =
        $.etqi = etq;
        etq ++;
        int i = 0;
        if(exps is Exp_una){
                etq += 3;
                etiqueta(exps.expresion)
                etq ++;
                i++;

        etq += 2;
        $.etqs = etq;

etiqueta(Delete(exp)) =
        $.etqi = etq;
        etiqueta(exp);
        etq++;
        $.etqs = etq;
```

etiqueta(New_cons(exp)) =
	$.etqi = etq;
	etiqueta(exp);
	etq += 2;
	$.etqs = etq;

etiqueta(Nl()) =
	$.etqi = etq;
	etq += 2;
	$.etqs = etq;

etiqueta(Write(exp)) =
	$.etqi = etq;
	etiqueta(exp);
	if (exesDesignador()()) =
		etq++;

	etq++;
	$.etqs = etq;

etiqueta(Read(exp)) =
	$.etqi = etq;
	etiqueta(exp);
	etq += 2;
	$.etqs = etq;

etiqueta(While_inst(exp, instrucciones)) =
	$.etqi = etq;
	etiqueta(exp);
	etq++;
	etiqueta(instrucciones);
	etq++;
	$.etqs = etq;

etiqueta(If_else (exp, instrucciones, instrucciones_else)) =
	if_else.etqi = etq;
	etiqueta(exp);
	etq++;
	etiqueta(instrucciones);
	etq++;
	etiqueta(instrucciones_else);
	if_else.etqs = etq;

etiqueta(If_inst(exp, instrucciones)) =
	$.etqi = etq;
	etiqueta(exp);
	etq++;
	etiqueta(instrucciones);

```
        $.etqs = etq;

etiqueta(Asig (exp0, exp1)) =
        $.etqi = etq;
        etiqueta(exp0);
        etiqueta(exp1);
        etq++;
        $.etqs = etq;

etiqueta(Insts_muchas (instrucciones, instruccion)) =
        etiqueta(instrucciones);
        etiqueta(instruccion);

etiqueta(Insts_una (instruccion)) =
        etiqueta(instruccion);

etiqueta(Lista_inst_una(instruccion)) =
        etiqueta(instruccion);

etiqueta(Lista_inst_muchas (instrucciones, instruccion)) =
        etiqueta(instrucciones);
        etiqueta(instruccion);

etiqueta(Bloque_prog(programa)) =
        etiqueta(programa);

etiqueta(Exp_muchas(expresiones, expresion)) =
        etiqueta(expresiones);
        etiqueta(expresion);

etiqueta(Exp_una(expresion)) =
        etiqueta(expresion);

etiqueta(Suma (arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
        etq++;
        $.etqs = etq;

etiqueta(Resta (arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
```

```
        etq++;
        $.etqs = etq;

etiqueta(And(arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
        etq++;
        $.etqs = etq;

etiqueta(Or (arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
        etq++;
        $.etqs = etq;

etiqueta(Menor (arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        if (arg0.gettipo == tNodo.BOOL) etq++;
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
        etq ++;
        $.etqs = etq;

etiqueta(Mayor (arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
        etq ++;
        $.etqs = etq;

etiqueta(MenorIgual (arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
        etq ++;
        $.etqs = etq;
```

etiqueta(MayorIgual (arg0, arg1)) =
```
$.etqi = etq;
etiqueta(arg0);
if (arg0.esDesignador()) etq++;
etiqueta(arg1);
if (arg1.esDesignador()) etq++;
etq ++;
$.etqs = etq;
```

etiqueta(Igual (arg0, arg1)) =
```
$.etqi = etq;
etiqueta(arg0);
if (arg0.esDesignador()) etq++;
etiqueta(arg1);
if (arg1.esDesignador()) etq++;
etq ++;
$.etqs = etq;
```

etiqueta(Distinto (arg0, arg1)) =
```
$.etqi = etq;
etiqueta(arg0);
if (arg0.esDesignador()) etq++;
etiqueta(arg1);
if (arg1.esDesignador()) etq++;
etq += 2;
$.etqs = etq;
```

etiqueta(Mul (arg0, arg1)) =
```
$.etqi = etq;
etiqueta(arg0)
if (arg0.esDesignador()) etq++;
etiqueta(arg1);
if (arg1.esDesignador()) etq++;
etq++;
$.etqs = etq;
```

etiqueta(Div (arg0, arg1)) =
```
$.etqi = etq;
exarg0().procesa(this);
if (arg0.esDesignador()) etq++;
exarg1().procesa(this);
if (arg1.esDesignador()) etq++;
etq++;
$.etqs = etq;
```

etiqueta(Percent (arg0, arg1)) =
```
$.etqi = etq;
```

```
        etiqueta(arg0)
        if (arg0.esDesignador()) etq++;
        etiqueta(arg1)
        if (arg1.esDesignador()) etq++;
        etq++;
        $.etqs = etq;

etiqueta(MenosUnario (arg0)) =
        $.etqi = etq;
        etq++;
        etiqueta(arg0);
        if (arg0.esDesignador()) etq++;
        etq++;
        $.etqs = etq;

etiqueta(Not (arg0)) =
        $.etqi = etq;
        etiqueta(arg0)
        if (arg0.esDesignador()) etq++;
        etq++;
        $.etqs = etq;

etiqueta(Corchete (arg0, arg1)) =
        $.etqi = etq;
        etiqueta(arg0);
        etiqueta(arg1);
        if (arg1.esDesignador()) etq++;
        etq += 3;
        $.etqs = etq;

etiqueta(Punto(exp)) =
        $.etqi = etq;
        etiqueta(exp)
        etq += 2;
        $.etqs = etq;

etiqueta(Flecha(exp)) =
        $.etqi = etq;
        etiqueta(exp)
        etq += 3;
        $.etqs = etq;

etiqueta(Star(arg0)) =
        $.etqi = etq
        etiqueta(arg0)
        etq++
        $.etqs = etq
```

etiqueta(True()) =
	$.etqi = etq;
	etq++;
	$.etqs = etq;

etiqueta(False()) =
	$.etqi = etq;
	etq++;
	$.etqs = etq;

etiqueta(LitReal()) =
	$.etqi = etq;
	etq++;
	$.etqs = etq;

etiqueta(Id(id)) =
	$.etqi = etq;
	if ($.nivel == 0)
		etq++;
	 else
		etq += 3;
	$.etqs = etq;

etiqueta(LitEnt()) =
	$.etqi = etq;
	etq++;
	$.etqs = etq;

etiqueta(LitNull()) =
	$.etqi = etq;
	etq++;
	$.etqs = etq;

etiqueta(LitCad()) =
	$.etqi = etq;
	etq++;
	$.etqs = etq;

6. Especificación del procesamiento de generación de código.

```
gen_cod(is_ninguna()) = skip

gen_cod(is_una(I)) =
        gen_cod(I)

gen_cod(is_muchas(Is,I)) =
        gen_cod(Is)
        gen_cod(I)

gen_cod(asig(E0,E1)) =
        gen_cod(E0)
        gen_cod(E1)
        gen_ins_asig(E1) // mueve o desapilaind, dependiendo de si
                         // E1 es o no un designador



gen_cod_params(pr_ninguno(),pf_ninguno()) = skip

gen_cod_params(pr_uno(P), pf_uno(PF)) =
        gen_cod_paso(P,PF)

gen_cod_params(pr_muchos(Ps,P), pf_muchos(PsF,PF)) =
        gen_cod_params(Ps,PsF)
        gen_cod_paso(P,PF)

gen_cod_paso(E,PF) =
        gen_ins(dup())
        gen_ins(apilaint(PF.dir))
        gen_ins(suma())
        gen_cod(E)
        gen_ins_paso(E,PF) //mueve si E no es designador, y PF es por
                           // valor, desapilaind en otro caso

gen_cod(num(n)) =
        gen_ins(apilaint(n))

gen_cod(id(v)) =
        si $.vinculo.nivel = 0
                gen_ins(apilaint($.vinculo.dir))
        si no
                gen_ins(apilad($.vinculo.nivel))
                gen_ins(apilaint($.vinculo.dir))
                gen_ins(suma())
                si $.vinculo = pf_var(T,v)
```

```
                    gen_ins(apilaind())

gen_cod(prim(E)) =
        gen_cod(E)

gen_cod(seg(E)) =
        gen_cod(E)
        sea ref!($.vinculo) = pair(T0,T1) en
                gen_ins(apilaint(T0.tam))
                gen_ins(suma())

gen_cod(dref(E)) =
        gen_cod(E)
        gen_ins(apilaint())

gen_cod(proc(id,PFs,Ds,Is)) =
        gen_cod(Is)
        gen_ins(desactiva($.nivel,$.tam))
        gen_ins(irind())
        recolecta_procs(Ds)
```

```
global procs = pila_vacia() // procedimientos pendientes de traducir

gen_cod(prog(Decs,Is)) =
        gen_cod(Is)
        recolecta_procs(Ds) // Este bucle provoca que se vayan traduciendo sucesivamente
                                // los procedimientos
        mientras(! es_vacia(procs)))
                P = pop(procs)
                gen_cod(P)

gen_cod(Decs_una(Dec)) =
        gen_cod(Dec)

gen_cod(Decs_muchas(Decs,Dec)) =
        gen_cod(Decs)
        gen_cod(Dec)

gen_cod(DecVar(T,id)) =
        skip

gen_cod(DecTipo(T,id)) =
        skip

gen_cod(DecProc(B)) =
        gen_cod(B)


gen_cod(Bloque_inst(B)) =
        gen_cod(B)

gen_cod(call(id,Ps)) =
        gen_ins(activa($.vinculo.nivel,$.vinculo.tam_datos,$.dir_sig))
        if ($.vinculo = proc(id,PsF,Ds,Is))
                gen_cod_params(Ps,PsF)
        gen_ins(desapilad($.vinculo.nivel))
        gen_ins(ir_a($.vinculo.dir_inic))

gen_cod(Delete (exp)) =
        gen_cod(exp)
        gen_ins(dealloc(extam_basico))

gen_cod(New_cons (exp)) =
        gen_cod(exp)
        gen_ins(alloc(extam_basico))
        gen_ins(desapilad())

gen_cod(Nl()) =
        gen_ins(apilaString("\n")
```

```
        gen_ins(writeString())

gen_cod(Write (exp)) =
        gen_cod(exp)
        if(exesDesignador())
                gen_ins(apilaInd())
        if(extipo == tNodo.LIT_ENT)
                gen_ins(writeInt())
        else if (extipo == tNodo.LIT_REAL)
                gen_ins(writeReal())
        else if (extipo == tNodo.BOOL)
                gen_ins(writeBooll())
        else if (extipo == tNodo.STRING)
                gen_ins(writeString())

gen_cod(Read (exp)) =
        gen_cod(exp)
        if(extipo == tNodo.LIT_ENT)
                gen_ins(writeInt())
        else if (extipo == tNodo.LIT_REAL)
                gen_ins(writeReal())
        else if (extipo == tNodo.STRING)
                gen_ins(writeString())
        gen_ins(desapilaInd())

gen_cod(While_inst (exp, instrucciones, etqi, etqs)) =
        gen_cod(exp)
        gen_ins(irF(etqs))
        gen_cod(instrucciones)
        gen_ins(irA(etqa))

gen_cod(If_else (exp, instrucciones, instrucciones_else, etqi, etqs)) =
        gen_cod(exp)
        gen_ins(irF(etqi))
        gen_cod(instrucciones)
        gen_ins(irA(etqs))
        gen_cod(instrucciones_else)

gen_cod(If_inst (exp, instrucciones)) =
        gen_cod(exp)
        gen_ins(irF(etqs))
        gen_cod(instrucciones)

gen_cod(Asig (exp0, exp1)) =
        gen_cod(exp0)
        gen_cod(exp1)
        if(exp1.esDesignador())
                gen_ins(mueve(exp1.tam_datos))
```

```
        else
                gen_ins(desapilaind())

gen_cod(Insts_muchas (instrucciones, instruccion)) =
        gen_cod(instrucciones)
        gen_cod(instruccion)

gen_cod(Insts_una (instruccion)) =
        gen_cod(instruccion)

gen_cod(Lista_inst_empty()) =
        skip

gen_cod(Lista_inst_una (instruccion)) =
        gen_cod(instruccion)

gen_cod(Lista_inst_muchas (instrucciones, instruccion)) =
        gen_cod(instrucciones)
        gen_cod(instruccion)

gen_cod(ParamForm (id, tipo, paramForm)) =
        skip

gen_cod(Pformal_ref (id, tipo, paramForm)) =
        skip

gen_cod(ParamForms_muchos (params, param)) =
        skip

gen_cod(ParamForms_uno (param)) =
        skip

gen_cod(ParamForms_empty()) =
        skip

gen_cod(Campos_muchos (campos, campo)) =
        skip

gen_cod(Campo_uno (campo)) =
        skip

gen_cod(Camp (tipo)) =
        skip

gen_cod(Bloque_prog (programa)) =
        gen_cod(programa)

gen_cod(No_bloque()) =
```

```
        skip

gen_cod(Lista_exp_empty()) =
        skip

gen_cod(Exp_muchas (expresiones, expresion))  =
        gen_cod(expresiones)
        gen_cod(expresion)

gen_cod(Exp_una (expresion)) =
        gen_cod(expresion)

gen_cod(Suma (arg0, arg1)) =
        gen_cod(arg0)
        if(arg0.esDesignador())
                gen_ins(apilaind())
        gen_cod(arg1)
        if(arg1.esDesignador())
                gen_ins(apilaind())
        if($.tipo  == tNodo.LIT_ENT)
                gen_ins(suma())
        else if($.tipo == tNodo.LIT_REAL)
                gen_ins(sumaR())

gen_cod(Resta (arg0, arg1)) =
        gen_cod(arg0)
        if(arg0.esDesignador())
                gen_ins(apilaind())
        gen_cod(arg1)
        if(arg1.esDesignador())
                gen_ins(apilaind())
        if($.tipo  == tNodo.LIT_ENT)
                gen_ins(resta())
        else if($.tipo == tNodo.LIT_REAL)
                gen_ins(restaR())

gen_cod(And (arg0, arg1)) =
        gen_cod(arg0)
        if(arg0.esDesignador())
                gen_ins(apilaind())
        gen_cod(arg1)
        if(arg1.esDesignador())
                gen_ins(apilaind())
        gen_ins(and())

gen_cod(Or (arg0, arg1)) =
        gen_cod(arg0)
        if(arg0.esDesignador())
```

```
            gen_ins(apilaind())
        gen_cod(arg1)
        if(arg1.esDesignador())
            gen_ins(apilaind())
        gen_ins(or())
```

gen_cod(Menor (arg0, arg1)) =

```
        gen_cod(arg0)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        gen_cod(arg1)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        if(arg0.tipo == tNodo.LIT_ENT)
            gen_ins(menor())
        else if(arg0.tipo == tNodo.LIT_REAL)
            gen_ins(menorR())
        else if(arg0.tipo == tNodo.STRING)
            gen_ins(menorString())
        else if(arg0.tipo == tNodo.BOOL)
            gen_ins(menorB())
```

gen_cod(Mayor (arg0, arg1)) =

```
        gen_cod(arg0)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        gen_cod(arg1)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        if(arg0.tipo == tNodo.LIT_ENT)
            gen_ins(mayor())
        else if(arg0.tipo == tNodo.LIT_REAL)
            gen_ins(mayorR())
        else if(arg0.tipo == tNodo.STRING)
            gen_ins(mayorString())
        else if(arg0.tipo == tNodo.BOOL)
            gen_ins(mayorB())
```

gen_cod(MenorIgual (arg0, arg1)) =

```
        gen_cod(arg0)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        gen_cod(arg1)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        if(arg0.tipo == tNodo.LIT_ENT)
            gen_ins(menorIg())
        else if(arg0.tipo == tNodo.LIT_REAL)
```

```
            gen_ins(menorIgR())
        else if(arg0.tipo == tNodo.STRING)
            gen_ins(menorIgString())
        else if(arg0.tipo == tNodo.BOOL)
            gen_ins(menorIgB())

gen_cod(MayorIgual (arg0, arg1)) =
        gen_cod(arg0)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        gen_cod(arg1)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        if(arg0.tipo == tNodo.LIT_ENT)
            gen_ins(mayorIg())
        else if(arg0.tipo == tNodo.LIT_REAL)
            gen_ins(mayorIgR())
        else if(arg0.tipo == tNodo.STRING)
            gen_ins(mayorIgString())
        else if(arg0.tipo == tNodo.BOOL)
            gen_ins(mayorIgB())

gen_cod(Igual (arg0, arg1)) =
        gen_cod(arg0)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        gen_cod(arg1)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        if(arg0.tipo == tNodo.LIT_ENT)
            gen_ins(Ig())
        else if(arg0.tipo == tNodo.LIT_REAL)
            gen_ins(IgR())
        else if(arg0.tipo == tNodo.STRING)
            gen_ins(IgString())
        else if(arg0.tipo == tNodo.BOOL)
            gen_ins(IgB())

gen_cod(Distinto (arg0, arg1)) =
        gen_cod(arg0)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        gen_cod(arg1)
        if(arg0.esDesignador())
            gen_ins(apilaind())
        if(arg0.tipo == tNodo.LIT_ENT)
            gen_ins(Ig())
        else if(arg0.tipo == tNodo.LIT_REAL)
```

```
            gen_ins(IgR())
        else if(arg0.tipo == tNodo.STRING)
            gen_ins(IgString())
        else if(arg0.tipo == tNodo.BOOL)
            gen_ins(IgB())
        gen_ins(not())

gen_cod(Mul (arg0, arg1)) =
        gen_cod(arg0)
        if (arg0.esDesignador()) gen_ins(apilaInd())
        gen_cod(arg1)
        if (arg1.esDesignador()) gen_ins(apilaInd())

        if (exp.tipo == tNodo.LIT_ENT)
            gen_ins(mul());
        else if (exp.tipo == tNodo.LIT_REAL)
            gen_ins(mulR());

gen_cod(Div (arg0, arg1)) =
        gen_cod(arg0)
        if (arg0.esDesignador()) gen_ins(apilaInd())
        gen_cod(arg1)
        if (arg1.esDesignador()) gen_ins(apilaInd())

        if (exp.tipo == tNodo.LIT_ENT)
            gen_ins(div());
        else if (exp.tipo == tNodo.LIT_REAL)
            gen_ins(divR());


gen_cod(Percent (arg0, arg1)) =
        gen_cod(arg0)
        if (arg0.esDesignador()) gen_ins(apilaInd())
        gen_cod(arg1)
        if (arg1.esDesignador()) gen_ins(apilaInd())
        gen_ins(percent());


gen_cod(MenosUnario (arg0)) =
        if ($.tipo == tNodo.LIT_ENT)
            gen_ins(apilaInt(0))
            gen_cod(arg0)
            if (arg0.esDesignador()) gen_ins(apilaInd())
            gen_ins(p.resta())
        else if (exp.gettipo == tNodo.LIT_REAL)
            gen_ins(apilaReal(0))
            gen_cod(arg0)
            if (arg0.esDesignador()) gen_ins(apilaInd())
```

```
            gen_ins(restaR())

gen_cod(Not (arg0)) =
        gen_cod(arg0)
        if (arg0().esDesignador()) gen_ins(apilaInd());
        gen_ins(not())

gen_cod(Corchete (arg0, arg1)) =
        gen_cod(arg0)
        gen_cod(arg1)
        if (arg1.esDesignador()) gen_ins(apilaInd());
        gen_ins(apilaInt(arg0.tam_basico))
        gen_ins(mul())
        gen_ins(suma())

gen_cod(Punto (exp)) =
        gen_cod(exp)
        gen_ins(apilaInt(exp.vinculo.desp))
        gen_ins(suma())

gen_cod(Flecha (exp)) =
        gen_cod(exp)
        gen_ins(apilaInd())
        gen_ins(apilaInt(exp.vinculo.desp))
        gen_ins(suma())

gen_cod(Star (arg0)) =
        gen_cod(arg0)
        gen_ins(apilaInd());

gen_cod(Id (id)) =
        if ($.nivel == 0)
                gen_ins(apilaInt(dir));
        else
                gen_ins(apilad(nivel));
                gen_ins(apilaInt(dir));
                gen_ins(suma());

gen_cod(True ()) =
        gen_ins(apilaBool(true));

gen_cod(False ()) =
        gen_ins(apilaBool(false));
```

gen_cod(LitEnt (num)) =
    gen_ins(apilaReal(num));

gen_cod(LitReal (num)) =
    gen_ins(apilaReal(num));

gen_cod(LitCad (cad)) =
    gen_ins(apilaString(cad));

gen_cod(LitNull ()) =
    gen_ins(apilaInt(-1));

gen_cod(Tipo_Id (id)) =
    skip

gen_cod(Array (tipo_array)) =
    skip

gen_cod(Record (campos)) =
    skip

gen_cod(Pointer (tipo)) =
    skip

gen_cod(Decs_una(Dec)) =
    gen_cod(Dec)

gen_cod(Decs_muchas(Decs,Dec)) =
    gen_cod(Decs)
    gen_cod(Dec)

gen_cod(int_cons())=
    skip

gen_cod(real_cons())=
    skip

gen_cod(bool_cons())=
    skip

gen_cod(string_cons())=
    skip