Grupo 14

- Félix García Ceballos
- Alberto Sánchez Gómez
- Vladimir Ustimenko

Sintaxis abstracta

Regla	Explicación
Prog_decs: Decs x Is-> Programa	El programa se compone de una lista de declaraciones y una lista de instrucciones.
Desc_vacia: -> Decs Decs_muchas: Decs x Dec -> Decs Decs_una: Dec -> Decs	Una lista de declaraciones puede o no tener declaraciones.
Dec_var: Tipo x String -> Dec	Construye una declaración de variable dado su tipo y su nombre
Dec_type: Tipo x String -> Dec	Construye una declaración de tipo dado su tipo y su nombre
Dec_proc: String x Param_forms x Decs x Is -> Dec	Construye una declaración de procedimiento dado su nombre, la lista de parámetros formales, la lista de declaraciones y la lista de instrucciones.
Pform_vacia: -> Param_forms Pform_uno: Param_form -> Param_forms Pform_muchos: Param_forms x Param_form -> Param_forms	Construye una lista de parámetros formales.
Pform: String x Tipo -> Param_form Pform_ref: String x Tipo -> Param_form	Construye un parámetro formal a partir de su nombre y su tipo.
int: -> Tipo	Tipo *int*
real: -> Tipo	Tipo *real*
bool: -> Tipo	Tipo *bool*
string: -> Tipo	Tipo *string*
ref: String -> Tipo	Nombre de otro tipo
array: String x Tipo -> Tipo	Tipo *array*
record: Campos -> Tipo	Estructura de dato

campos_uno: Campo -> Campos campos_muchos: Campos x Campo -> Campos	Constructor de lista de campos.
campo: Tipo x String -> Campo	Constructor campo
pointer: Tipo -> Tipo	Contruye un tipo puntero, dado el tipo base
is_muchas: Is x I -> Is	Construye una lista de instrucciones a partir de otra lista de instrucciones y una instrucción.
is_una: I -> Is	Construye una lista de instrucciones a partir de una instrucción.
is_vacia: -> Is	Construye lista de instrucciones vacía.
exp_vacio: -> Exps exp_uno: Exp -> Exps exp_muchos: Exps x Exp -> Exps	Construye lista de expresiones.
asig: Exp x Exp -> I	Construye una instrucción a partir de la expresión que representa la parte izquierda, y la expresión que representa la parte derecha.
if_then: Exp x Is -> I	Instrucción de salto if
if_then_else: Exp x Is x Is -> I	Instrucción de salto if-then-else
while: Exp x Is -> I	Instruccion de while
read: Exp -> I	Instrucción de read
write: Exp -> I	Instrucción de write
nl: -> I	Nueva línea
new: Exp -> I	Instrucción de new
delete: Exp -> I	Instrucción de delete
call: Exp x Param_real -> I	Invocación de una función, a partir de su expresión y la lista de sus parámetros.
Preal_uno: Param_real -> Param_reals	Crea una lista de parámetros a partir de un único parámetro.
Preal_muchos: Param_reals x Param_real -> Param_reals	Crea una lista de parámetros a partir de otra lista de parámetros.
i_comp: Decs x Is -> I	Instrucción compuesta.

lit_int: String -> Exp	Literal entero
lit_real: String -> Exp	Literal real
lit_str: String -> Exp	Literal cadena
true: -> Exp	True
false: -> Exp	False
id: string -> Exp	Construye una expresión que representa una variable a partir de su nombre.
null: -> Exp	Expresión *null*
menor: Exp x Exp -> Exp mayor: Exp x Exp -> Exp menor_igual: Exp x Exp -> Exp mayor_igual: Exp x Exp -> Exp igual: Exp x Exp -> Exp distinto: Exp x Exp -> Exp	Operadores relacionales
suma: Exp x Exp -> Exp resta: Exp x Exp -> Exp mul: Exp x Exp -> Exp div: Exp x Exp -> Exp modulo: Exp x Exp -> Exp and: Exp x Exp -> Exp or: Exp x Exp -> Exp	Expresiones compuestas
negativo: Exp -> Exp not: Exp -> Exp	Negación 1-aria y negación lógica.
acc: Exp x string -> Exp	Acceso al campo (2do argumento) de un registro (1er argumento)
indx: Exp x Exp -> Exp	Indexación de un elemento (2do argumento) en un array (1er argumento)
dref: Exp -> Exp	Valor apuntado por un puntero (dereferencia)

Procesamiento de vinculación

```
global ts // tabla de símbolos

vincula(prog(Decs, Is)) =
        if(!Decs.is_empty())
                vincula1(Decs)
                vincula2(Decs)
        vincula(Is)

vincula1(Decs_vacia()) =
        skip

vincula1(Decs_una(Dec)) =
        vincula1(Dec)

vincula1(Decs_muchas(Decs, Dec)) =
        vincula1(Decs)
        vincula1(Dec)

vincula1(Dec_var(T, id)) =
        vincula1(T)
        recolecta(id, $)

vincula1(dec_tipo(T,id)) =
        vincula1(T)
        recolecta(id,$)

vincula1(dec_proc(id,Ps,Ds,Is)) =
        recolecta(id,$)
        abre_nivel(ts) // se abre el nivel del procedimiento en la tabla de símbolos
        vincula1(Ps)
        vincula1(Ds)
        vincula2(Ps)
        vincula2(Ds)
        vincula_procs(Ds)
        vincula(Is)
        cierra_nivel(ts) // se elimina el nivel del procedimiento

vincula1(Pform_vacia()) =
        skip

vincula1(Pform_una(Pform)) =
        vincula1(Pform)
```

```
vincula1(Pform_muchas(Pforms, Pform)) =
        vincula1(Pforms)
        vincula1(Pform)

vincula1(Pform(id,T)) =
        vincula1(T)
        recolecta(id, $)

vincula1(Pform_ref(id, T)) =
        vincula1(T)
        recolecta(id, $)

vincula1(int())=
        skip //nada que vincular

vincula1(real())=
        skip //nada que vincular

vincula1(bool())=
        skip //nada que vincular

vincula1(string())=
        skip //nada que vincular

vincula1(ref(id)) =
        si existe_id(id,ts) entonces
                $.vinculo = valorDe(ts,id) // El tipo está declarado: se vincula
        si no
                error // tipo no declarado

vincula1(array(T, id)) =
        vincula1(T)

vincula1(record(Cs)) =
        vincula1(Cs)

vincula1(campo_uno(C)) =
        vincula1(C)

vincula1(campo_muchos(Cs, C)) =
        vincula1(Cs)
        vincula1(C)

vincula1(campo(T, id)) =
        vincula1(T)
        recolecta(id, $)

vincula1(pair(T1,T2)) =
```

```
        vincula1(T1)
        vincula1(T2)

vincula1(pointer(T)) =
        si T /=  ref(_) entonces // Se vincula únicamente si no se trata de un “pointer – ref”
        vincula1(T)

vincula1(is_vacia()) =
        skip

vincula1(is_una(I)) =
        vincula1(I)

vincula1(is_muchas(Is, I)) =
        vincula1(Is)
        vincula1(I)

vincula1(exp_vacio()) =
        skip

vincula1(exp_uno(E)) =
        vincula1(E)

vincula1(exp_muchos(Es, E)) =
        vincula1(Es)
        vincula1(E)

vincula1(asig(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(if_then(E, Is)) =
        vincula1(E)
        vincula1(Is)

vincula1(if_then_else(E, Is1, Is2)) =
        vincula1(E)
        vincula1(Is1)
        vincula1(Is2)

vincula1(while(E, Is)) =
        vincula1(E)
        vincula1(Is)

vincula1(read(E)) =
        vincula1(E)

vincula1(write(E)) =
```

```
        vincula1(E)

vincula1(nl()) =
        skip

vincula1(new(E)) =
        vincula1(E)

vincula1(delete(E)) =
        vincula1(E)

vincula1(call(id,Ps)) =
        si existe_id(ts, id) entonces
                $.vinculo = valorDe(ts,id)
        si no
                error // id no declarado
        vincula(Ps)

 vincula1(Preal_uno(E)) =
         vincula(E)

vincula1(Preal_muchos(Ps,E)) =
        vincula(Ps)
        vincula(E)

vincula1(i_comp(Ds, Is)) =
        vincula1(Ds)
        vincula1(Is)

vincula1(lit_int(id)) =
        skip

vincula1(lit_real(id)) =
        skip

vincula1(lit_str(id)) =
        skip

vincula1(true()) =
        skip

vincula1(false()) =
        skip

vincula1(id(id)) =
        si existe_id(ts, id) entonces
                $.vinculo = valorDe(ts,id)
        si no
```

```
        error // id no declarado

vincula1(null()) =
        skip

vincula1(menor(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(mayor(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(menor_igual(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(mayor_igual(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(igual(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(distinto(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(suma(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(resta(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(mul(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(div(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(modulo(E1,E2)) =
        vincula1(E1)
        vincula1(E2)
```

```
vincula1(and(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(or(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(negativo(E)) =
        vincula1(E)

vincula1(not(E)) =
        vincula1(E)

vincula1(acc(E,c)) =
        vincula1(E)

vincula1(indx(E1,E2)) =
        vincula1(E1)
        vincula1(E2)

vincula1(dref(E)) =
        vincula1(E)



// Segunda pasada

vincula2(decs_ninguna()) =
        skip

vincula2(decs_una(Dec)) =
        vincula2(Dec)

vincula2(decs_muchas(Decs,Dec)) =
        vincula2(Decs)
        vincula2(Dec)

vincula2(dec_var(T,id)) =
        vincula2(T)

vincula2(dec_tipo(T,id)) =
        vincula2(T)

vincula2(dec_proc(id,Ps,Ds,Is)) =
        skip
```

```
vincula2(ref(id)) =
        skip

vincula2(int())=
        skip

vincula2(real())=
        skip

vincula2(bool())=
        skip

vincula2(string())=
        skip

vincula2(array(T,n)) =
        vincula2(T)

vincula2(record(Cs)) =
        vincula2(Cs)

vincula2(campos_muchos(Cs,C)) =
        vincula2(Cs)
        vincula2(C)

vincula2(campos_uno(C)) =
        vincula2(C)

vincula2(campo(T, id)) =
        vincula2(T)

vincula2(pair(T1,T2)) =
        vincula2(T1)
        vincula2(T2)

vincula2(Pform_vacia()) =
        skip

vincula2(Pform_una(Pform)) =
        vincula2(Pform)


vincula2(Pform_muchas(Pforms, Pform)) =
        vincula2(Pforms)
        vincula2(Pform)

vincula2(Pform(id,T)) =
        vincula2(T)
```

```
vincula2(Pform_ref(id, T)) =
        vincula2(T)

vincula2(pointer(T)) =
        si T = ref(id) entonces // Se vincula el ref(id)
                si existe_id(id,ts) entonces
                        T.vinculo = valorDe(ts,id) // El id referido está declarado: se vincula
                si no
                        error // tipo no declarado
        si no
                vincula2(T)


recolecta(id, Dec) =
        si id_duplicado(ts,id) entonces
                error // id duplicado.
        si no
                añade(ts, id, Dec) // Se añade una entrada a la tabla de símbolos
                                   // que asocia la declaración al identificador
```

Comprobación de tipado

```
//OPERADORES ARITMETICOS

chequeo tipo(prog(Ds, Is)) =
        chequeo_tipo(Is)
        $.tipo = Is.tipo
chequeo_tipo(is_muchas(Is, I)) =
        chequeo_tipo(Is)
        chequeo_tipo(I)
        si Is.tipo = ok() y I.tipo = ok entonces
                $.tipo = ok
        else
                $.tipo = error()

chequeo_tipo(is_una(I) =
        chequeo_tipo(I)
        $.tipo = I.tipo

chequeo_tipo(asig(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si son_compatibles(e0.tipo, e1.tipo) entonces
                $.tipo = ok()
```

```
        si no
                error //Si los tipos no son compatibles y ninguno es tipo error, devolver error.
                $.tipo = error()

chequeo_tipo(id(id)) =
        sea $.vinculo = Dec en
                si Dec = dec_var(T, _) entonces
                        $.tipo = T
                si no
                        error //El id no es una variable.
                        $.tipo = error()

chequeo_tipo(suma(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = int() y ref!(e1) = int() entonces
                $.tipo = int()
        si ref!(e0) = int() y ref!(e1) = real() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = int() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = real() entonces
                        $.tipo = real()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

chequeo_tipo(resta(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = int() y ref!(e1) = int() entonces
                $.tipo = int()
        si ref!(e0) = int() y ref!(e1) = real() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = int() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = real() entonces
                        $.tipo = real()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

chequeo_tipo(mul(e0, e1)) =
        chequeo_tipo(e0)
```

```
        chequeo_tipo(e1)
        si ref!(e0) = int() y ref!(e1) = int() entonces
                $.tipo = int()
        si ref!(e0) = int() y ref!(e1) = real() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = int() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = real() entonces
                        $.tipo = real()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

chequeo_tipo(div(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = int() y ref!(e1) = int() entonces
                $.tipo = int()
        si ref!(e0) = int() y ref!(e1) = real() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = int() entonces
                        $.tipo = real()
        si ref!(e0) = real() y ref!(e1) = real() entonces
                        $.tipo = real()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

//OPERADOR MODULO ENTERO

chequeo_tipo(modulo(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = int() y ref!(e1) = int() entonces
                $.tipo = int()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

//OPERADORES BOOLEANOS

chequeo_tipo(and(e0, e1)) =
```

```
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
                $.tipo = bool()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()



chequeo_tipo(or(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
                $.tipo = bool()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

//OPERADORES RELACIONALES

chequeo_tipo(menor(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
                $.tipo = bool()
                //Valores booleanos se consideran comparables
        si ref!(e0) = (int() o real()) y ref!(e1) = (int() o real()) entonces
                $.tipo = bool()
                //Se pueden comparar enteros con reales
        si ref!(e0) = string() y ref!(e1) = string() entonces
                $.tipo = bool()
                //Se pueden comparar strings alfabéticamente
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()



chequeo_tipo(mayor(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
```

```
                $.tipo = bool()
                //Valores booleanos se consideran comparables
        si ref!(e0) = (int() o real()) y ref!(e1) = (int() o real()) entonces
                $.tipo = bool()
                //Se pueden comparar enteros con reales
        si ref!(e0) = string() y ref!(e1) = string() entonces
                $.tipo = bool()
                //Se pueden comparar strings alfabéticamente
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

chequeo_tipo(menor_igual(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
                $.tipo = bool()
                //Valores booleanos se consideran comparables
        si ref!(e0) = (int() o real()) y ref!(e1) = (int() o real()) entonces
                $.tipo = bool()
                //Se pueden comparar enteros con reales
        si ref!(e0) = string() y ref!(e1) = string() entonces
                $.tipo = bool()
                //Se pueden comparar strings alfabéticamente
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

chequeo_tipo(mayor_igual(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
                $.tipo = bool()
                //Valores booleanos se consideran comparables
        si ref!(e0) = (int() o real()) y ref!(e1) = (int() o real()) entonces
                $.tipo = bool()
                //Se pueden comparar enteros con reales
        si ref!(e0) = string() y ref!(e1) = string() entonces
                $.tipo = bool()
                //Se pueden comparar strings alfabéticamente
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
```

```
                $.tipo = error()

chequeo_tipo(igual(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
                $.tipo = bool()
                //Valores booleanos se consideran comparables
        si ref!(e0) = (int() o real()) y ref!(e1) = (int() o real()) entonces
                $.tipo = bool()
                //Se pueden comparar enteros con reales
        si ref!(e0) = string() y ref!(e1) = string() entonces
                $.tipo = bool()
                //Se pueden comparar strings alfabéticamente
        si ref!(e0) = (pointer() o null) y ref!(e1) = (pointer() o null) entonces
                $.tipo = bool()
                //Se pueden comparar punteros con null y entre sí, independientemente de
                los tipos a los que referencien.
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()

chequeo_tipo(distinto(e0, e1)) =
        chequeo_tipo(e0)
        chequeo_tipo(e1)
        si ref!(e0) = bool() y ref!(e1) = bool() entonces
                $.tipo = bool()
                //Valores booleanos se consideran comparables
        si ref!(e0) = (int() o real()) y ref!(e1) = (int() o real()) entonces
                $.tipo = bool()
                //Se pueden comparar enteros con reales
        si ref!(e0) = string() y ref!(e1) = string() entonces
                $.tipo = bool()
                //Se pueden comparar strings alfabéticamente
        si ref!(e0) = (pointer() o null) y ref!(e1) = (pointer() o null) entonces
                $.tipo = bool()
                //Se pueden comparar punteros con null y entre sí, independientemente de
                los tipos a los que referencien.
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que ninguno de los
                tipos sea ya error.
                $.tipo = error()
```

```
//EXPRESIONES TIPO *E - NOT Y NEGATIVO

chequeo_tipo(negativo(E)) =
        chequeo_tipo(E)
        si ref!(E) = int()
                $.tipo = int()
        si ref!(E) = real()
                $.tipo = real()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que el tipo no sea
                error.
                $.tipo = error()


chequeo_tipo(not(E)) =
        chequeo_tipo(E)
        si ref!(E) = bool()
                $.tipo = bool()
        si no
                error
                //Informar de error de compatibilidad de tipos, siempre que el tipo no sea
                error.
                $.tipo = error()

//EXPRESIONES TIPO E* - ACCESO, INDEXACIÓN Y DE-REFERENCIA

chequeo_tipo(acc(E)) =
        chequeo_tipo(E)
        si ref!(E.tipo) = record(Cs) y existeCampo(Cs, c) entonces
                $.tipo = tipoDeCampo(Cs, c)
        si no
                error
                // Informar del error que se ha producido (el tipo de E no es un registro, el
                campo no existe) siempre que el tipo no sea ya error.
                $.tipo = error()

chequeo_tipo(indx(E0, E1)) =
        chequeo_tipo(E0)
        chequeo_tipo(E1)
        si ref!(E0.tipo) = array(T,_) y ref!(E1.tipo) = int entonces
                $.tipo = T
        si no
                error
                // Informar del error que se ha producido (E0 no es error pero no es un array,
                o E0 es un array y E1 no es un error, pero tampoco un entero) siempre que
                los tipos no sean error.
                $.tipo = error()
```

```
chequeo_tipo(dref(E)) =
        chequeo_tipo(E)
        si ref!(E.tipo) = pointer(T) entonces
                $.tipo = T
        si no
                error
                // Informar del error que se ha producido, siempre que el tipo de T no sea ya
                error
                $.tipo = error()
```

Asignación de espacio

```
global dir=0
global nivel=0
# Definimos el tamaño de capa tipo. Lo guardamos en tam
asigna_espacio_tipo(T) =
si indefinido(T.tam) {
        si(T == int){
                T.tam = 1
        }
        si(T == bool){
                T.tam = 1
        }
        si(T == real){
                T.tam = 1
        }
        si(T == bool){
                T.tam = 1
        }
        si(T == string){
                T.tam = 1
        }
        si(T == pointer){
                T.tam = 1 # un puntero es siempre de tamaño 1
        }
        else{

        }


}
```

Repertorio instrucciones máquina P

Instrucciones aritmético-lógicas	
suma resta mul div modulo and or menor mayor menor_igual mayor_igual igual distinto negativo not	Desapila los argumentos de la pila de evaluación (los argumentos aparecen en la pila en orden inverso; por ejemplo, si la operación necesita dos argumentos, en la cima estará el 2º argumento, en la sub-cima el 1er argumento) , realizan la operación , y apilan el resultado en la pila de evaluación.
negativo	Desapila la cima de la pila y apila el valor con el signo opuesto
not	Desapila la cima de la pila y apila la negación lógica
Instrucciones de movimiento de datos	
apilaint(v) apilareal(v) apilabool(v) apilastring(v) apilanull	Apilan el valor v en la pila de evaluación, una instrucción por cada tipo básico t en el lenguaje
apilaind	Desapila una dirección d de la pila de evaluación, y apila en dicha pila el contenido de la celda d en la memoria de datos
desapilaind	Desapila un valor v y una dirección d de la pila de evaluación (primero v, después d), y actualiza el contenido de la celda d en la memoria de datos a v.
mueve(n)	Desapila dos direcciones d1 y d0 de la pila de evaluación (primero d1, luego d0). Seguidamente copia el contenido de las n

	celdas consecutivas que comienzan en la dirección d1 a las correspondientes n celdas que comienzan en la dirección d0.
Instrucciones de salto	
ira(d)	Salta incondicionalmente a la dirección d
irf(d)	Desapila un valor v de la pila de evaluación. Si es falso salta a d. Si no, continúa en secuencia.
irv(d)	Desapila un valor v de la pila de evaluación. Si es cierto salta a d. Si no, continúa en secuencia.
irind	Desapila una dirección d de la pila de evaluación, y realiza un salto incondicional a dicha dirección.
Gestión de memoria dinámica	
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 n.
Soporte a la ejecución de procedimientos	
activa(n,t,d)	Reserva espacio en el segmento de pila de registros de activación para ejecutar un procedimiento que tiene nivel de anidamiento n y tamaño de datos locales t. Así mismo, almacena en la zona de control de dicho registro d como dirección de retorno. También almacena en dicha zona de control el valor del display de nivl n. Por último, apila en la pila de evaluación la dirección de comienzo de los datos en el registro creado.
apilad(n)	Apila en la pila de evaluación el valor del display de nivel n
desapilad(n)	Desapila una dirección d de la pila de evaluación en el display de nivel n.

desactiva(n,t)	Libera el espacio ocupado por el registro de activación actual, restaurando adecuadamente el estado de la máquina. n indica el nivel de anidamiento del procedimiento asociado; t el tamaño de los datos locales. De esta forma, la instrucción: (i) apila en la pila de evaluación la dirección de retorno; (ii) restaura el valor del display de nivel n al antiguo valor guardado en el registro; (iii) decrementa el puntero de pila de registros de activación en el tamaño ocupado por el registro.
dup	Consulta el valor v de la cima de la pila de evaluación, y apila de nuevo dicho valor (es decir, duplica la cima de la pila de evaluación)
stop	Detiene la máquina
Instrucciones para escritura y lectura	
writeint(v) writereal(v) writebool(v) writestring(v)	Desapila un valor de la pila de evaluación, y escribe dicho valor en la cinta de salida
readint readreal readbool readstring	Lee un valor de la cinta de entrada y lo apila en la pila de evaluación

Etiquetado

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

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

```
etiqueta(Decs_vacia()) =
        skip

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

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

etiqueta(dec_proc(id,Ps,Ds,Is)) =
        $.inic = etq
        etiqueta(Is)
        etq = etq + 2
        recolecta_procs(Ds)
        $.sig = etq


etiqueta(is_vacia()) =
        skip

etiqueta(is_una(I)) =
        etiqueta(I)

etiqueta(is_muchas(Is, I)) =
        etiqueta(Is)
        etiqueta(I)

etiqueta(exp_vacio()) =
        skip

etiqueta(exp_uno(E)) =
        etiqueta(E)

etiqueta(exp_muchos(Es, E)) =
        etiqueta(Es)
        etiqueta(E)

etiqueta(asig(E1,E2)) =
        $.inic = etq
        etiqueta(E1)
        etiqueta(E2)
        etq++
        $.sig = etq

etiqueta(if_then(E, Is)) =
        $.inic = etq
        etiqueta(E)
```

```
        etq++
        etiqueta(Is)
        $.sig = etq

etiqueta(if_then_else(E, Is1, Is2)) =
        $.inic = etq
        etiqueta(E)
        etq++
        etiqueta(Is1)
        etq++
        etiqueta(Is2)
        $.sig = etq

etiqueta(while(E, Is)) =
        $.inic = etq
        etiqueta(E)
        etq++
        etiqueta(Is)
        etq++
        $.sig = etq

etiqueta(read(E)) =
        $.inic = etq
        etiqueta(E)
        etq = etq + 2
        $.sig = etq

etiqueta(write(E)) =
        $.inic = etq
        etiqueta(E)
        etq++
        $.sig = etq


etiqueta(nl()) =
        $.inic = etq
        etq = etq + 2
        $.sig = etq

etiqueta(new(E)) =
        $.inic = etq
        etiqueta(E)
        etq = etq + 2
        $.sig = etq

etiqueta(delete(E)) =
        $.inic = etq
        etiqueta(E)
```

```
        etq++
        $.sig = etq

etiqueta(call(id,Ps)) =
        $.inic = etq
        etq++
        sea $.vinculo = proc(id, PsF, Ds, Is) en
                etiqueta_params(Ps,PsF)
        fin sea
        etq = etq + 2
        $.sig = etq

etiqueta(i_comp(Ds, Is)) =
        etiqueta(Ds)
        etiqueta(Is)

etiqueta(lit_int(id)) =
        etq++

etiqueta(lit_real(id)) =
        etq++

etiqueta(lit_str(id)) =
        etq++

etiqueta(true()) =
        etq++

etiqueta(false()) =
        etq++

etiqueta(id(id)) =
        si $.vinculo.nivel = 0
                etq++
        si no
                etq = etq + 3
                si $.vinculo = pf_var(T,v)
                        etq++


etiqueta(null()) =
        etq++

etiqueta(menor(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
```

```
        if(E2.esDesignador())
                etq++
        etq++

etiqueta(mayor(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(menor_igual(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(mayor_igual(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(igual(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(distinto(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
```

```
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(suma(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(resta(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(mul(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(div(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(modulo(E1,E2)) =
        etiqueta(E1)
```

```
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(and(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(or(E1,E2)) =
        etiqueta(E1)
        if(E1.esDesignador())
                etq++
        etiqueta(E2)
        if(E2.esDesignador())
                etq++
        etq++


etiqueta(negativo(E)) =
        etiqueta(E)
        if(E.esDesignador())
                etq++
        etq++


etiqueta(not(E)) =
        etiqueta(E)
        if(E.esDesignador())
                etq++
        etq++

etiqueta(acc(E,c)) = (No se?)

etiqueta(indx(E1,E2)) =
        etiqueta(E1)
        etiqueta(E2)
        si E1.esDesignador()
                etq = etq +4
```

```
etiqueta(dref(E)) =
        etiqueta(E)
        etq++

etiqueta_params(pr_ninguno(),pf_ninguno()) =
        skip
etiqueta_params(pr_uno(P), pf_uno(PF)) =
        etiqueta_paso(P,PF)

etiqueta_params(pr_muchos(Ps,P), pf_muchos(PsF,PF)) =
        etiqueta_params(Ps,PsF)
        etiqueta_paso(P,PF)

etiqueta_paso(E,PF) =
        etq = etq + 3
        etiqueta(E)
        etq++
```

Generación de código

```
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_vacia()) =
        skip

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

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

gen_cod(dec_proc(id,Ps,Ds,Is)) =
        gen_cod(Is)
        gen_ins(desactiva($.nivel, $.tam))
```

```
        gen_ins(irind())
        recolecta_procs(Ds)


gen_cod(is_vacia()) =
        skip

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

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

gen_cod(exp_vacio()) =
        skip

gen_cod(exp_uno(E)) =
        gen_cod(E)

gen_cod(exp_muchos(Es, E)) =
        gen_cod(Es)
        gen_cod(E)

gen_cod(asig(E1,E2)) =
        gen_cod(E1)
        gen_cod(E2)
        if(E1.esDesignador())
                gen_ins(mueve(E1.tam))
        else
                gen_ins(desapilaind())

gen_cod(if_then(E, Is)) =
        gen_cod(E)
        gen_ins(irf($.sig)
        gen_cod(Is)

gen_cod(if_then_else(E, Is1, Is2)) =
        gen_cod(E)
        gen_ins(irf(Is2.inic)
        gen_cod(Is1)
        gen_ins(ira($.sig))
        gen_cod(Is2)

gen_cod(while(E, Is)) =
        gen_cod(E)
        gen_ins(irf($.sig))
        gen_cod(Is)
```

```
        gen_ins(ira($.inic))

gen_cod(read(E)) =
        gen_cod(E)
        if(E.tipo() == int)
                gen_ins(readint())
        else if(E.tipo() == real)
                gen_ins(readreal())
        else if(E.tipo() == string)
                gen_ins(readstring())
        else if(E.tipo() == bool)
                gen_ins(readbool())
        gen_ins(desapilaind())

gen_cod(write(E)) =
        gen_cod(E)
        if(E.tipo() == int)
                gen_ins(writeint())
        else if(E.tipo() == real)
                gen_ins(writereal())
        else if(E.tipo() == string)
                gen_ins(writestring())
        else if(E.tipo() == bool)
                gen_ins(writebool())


gen_cod(nl()) =
        gen_ins(apilastring("\n"))
        gen_ins(writestring())

gen_cod(new(E)) =
        gen_cod(E)
        gen_ins(alloc(E.tam_base))
        gen_ins(desapilaind())

gen_cod(delete(E)) =
        gen_cod(E)
        gen_ins(dealloc(E.tam_base))

gen_cod(call(id,Ps)) =
        gen_ins(activa($.vinculo.nivel, $.vinculo.tam_datos, $.sig))
        sea $.vinculo = proc(id, PsF, Ds, Is) en
                gen_cod_params(Ps,PsF)
        fin sea
        gen_ins(desapilad($.vinculo.nivel))
        gen_ins(ir_a($.vinculo.inic))

gen_cod(i_comp(Ds, Is)) =
```

```
        gen_cod(Ds)
        gen_cod(Is)

gen_cod(lit_int(id)) =
        gen_ins(apilaint(id))

gen_cod(lit_real(id)) =
        gen_ins(apilareal(id))

gen_cod(lit_str(id)) =
        gen_ins(apilastring(id))

gen_cod(true()) =
        gen_ins(apilabool(true))

gen_cod(false()) =
        gen_ins(apilabool(false))

gen_cod(id(id)) =
        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(null()) =
        gen_ins(apilanull)

gen_cod(menor(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(menor())

gen_cod(mayor(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
```

```
        gen_ins(mayor())

gen_cod(menor_igual(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(menor_igual())

gen_cod(mayor_igual(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(mayor_igual())

gen_cod(igual(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(igual())

gen_cod(distinto(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(distinto())

gen_cod(suma(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(suma())

gen_cod(resta(E1,E2)) =
```

```
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(resta())

gen_cod(mul(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(mul())

gen_cod(div(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(div())

gen_cod(modulo(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(modulo())

gen_cod(and(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(and())

gen_cod(or(E1,E2)) =
        gen_cod(E1)
        if(E1.esDesignador())
                gen_ins(apilaind()) // para obtener el valor si es un designador
```

```
        gen_cod(E2)
        if(E2.esDesignador())
                gen_ins(apilaind())
        gen_ins(or())

gen_cod(negativo(E)) =
        gen_cod(E)
        if(E.esDesignador())
                gen_ins(apilaind())
        gen_ins(negativo())


gen_cod(not(E)) =
        gen_cod(E)
        if(E.esDesignador())
                gen_ins(apilaind())
        gen_ins(not())

gen_cod(acc(E,c)) = (No se?)

gen_cod(indx(E1,E2)) =
        gen_cod(E1)
        gen_cod(E2)
        si E1.esDesignador()
                gen_ins(apilaind())
        sea E0.tipo = array(n,T)
                gen_ins(apilaint(T.tam)
                gen_ins(mul)
                gen_ins(suma)

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

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)
```

```
        if (!E.esDesignador() && PF.esValor())
                gen_ins(mueve(E.tam))
        else
                gen_ins(desapilaind())


recolecta_procs(decs_ninguna()) = skip

recolecta_procs(decs_una(Dec)) = recolecta_procs(Dec)

recolecta_procs(decs_muchas(Decs,Dec)) =
        recolecta_procs(Decs)
        recolecta_procs(Dec)

recolecta_procs(dec_var(T,id)) = skip

recolecta_procs(dec_tipo(T,id)) = skip

recolecta_procs(dec_proc(id,Ps,Ds,Is)) =
        push($,procs) // se añade el procedimiento a la pila de procedimientos pendientes de
traducción
```