

## **Práctica Traductores de Lenguajes**

G-Compilador

**Andrés Ollero Morales**  
**Víctor Alejandro Sanz Ararat**  
**Gabriel de Oliveira Trindade**



Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software  
Escuela Técnica Superior De Ingenieros Informáticos  
Universidad Politécnica de Madrid  
Junio 2023

# Índice

<b>1. Diseño final del Procesador</b>	<b>2</b>
1.1. Diseño de Tokens . . . . .	2
1.2. Gramática de Contexto Libre (Tipo II) . . . . .	2
<b>2. Diseño del Generador de Código Intermedio</b>	<b>4</b>
2.1. Traducción Dirigida Por la Sintáxis con Acciones Semánticas . . . . .	4
<b>3. Diseño del Generador de Código Final</b>	<b>8</b>
3.1. Plantilla de Traducción de Cuartetos . . . . .	8
3.2. Tratamiento de las Strings . . . . .	10
<b>4. Diseño del Entorno de Ejecución</b>	<b>10</b>
4.1. Esquema de la memoria en ejecución . . . . .	10
4.2. Diseño del Registro de Activación . . . . .	11
<b>A. Casos de Prueba</b>	<b>12</b>
A.1. Prueba Funcional 1 . . . . .	12
A.2. Prueba Funcional 2 . . . . .	14
A.3. Prueba Funcional 3 . . . . .	16
A.4. Prueba Funcional 4 . . . . .	19
A.5. Prueba Funcional 5 . . . . .	22

## 1. Diseño final del Procesador

A modo de introducción, hemos querido recordar lo que es capaz de procesar el compilador. La implementación fue realizada en el lenguaje de programación Java. Nuestro grupo contaba con las siguientes opciones de práctica:

- Técnicas de Análisis Sintáctico: Descendente con tabla
- Sentencias: Sentencia de selección múltiple (switch-case)
- Operadores especiales: Asignación con resto (%=)
- Comentarios: Comentario de bloque (/\* \*/)
- Cadenas: Con comillas dobles ("" )

### 1.1. Diseño de Tokens

<palabraReservada, boolean>	<palabraReservada, input>
<palabraReservada, break>	<palabraReservada, int>
<palabraReservada, case>	<palabraReservada, let>
<palabraReservada, function>	<palabraReservada, switch>
<palabraReservada, print>	<palabraReservada, return>
<palabraReservada, string>	<palabraReservada, if>
<palabraReservada, default>	

<suma, >	<puntoComa, >	<asignacion, >
<negacion, >	<coma, >	<dosPuntos, >
<abrePar, >	<cierraPar, >	<comparacion, >
<abreLlave, >	<cierraLlave, >	<asignacionResto, >
<id, n <sup>º</sup> TS>	<constEnt, n <sup>º</sup> >	<cadena, "lexema">

### 1.2. Gramática de Contexto Libre (Tipo II)

Esta gramática es sobre la que hacemos las acciones semánticas mediante Traducción Dirigida por la Sintáxis.

Axioma = PP

NoTerminales = { PP P S SS E R RR U UU V VV L Q X B T A K C F H O D }

Terminales = { ! == + id ( ) constEnt cadena %= print input return , if break  
switch case int boolean string let function ; : = { } default }

Producciones = {  
E -> R RR  
RR -> == R RR  
RR -> lambda  
R -> U UU  
UU -> + U UU  
UU -> lambda

```

U -> ! V
U -> V
V -> id VV
V -> ( E )
V -> constEnt
V -> cadena
VV -> ( L )
VV -> lambda
S -> id SS
SS -> %= E ;
SS -> = E ;
SS -> ( L ) ;
S -> print R ;
S -> input id ;
S -> return X ;
L -> E Q
L -> lambda
Q -> , E Q
Q -> lambda
X -> E
X -> lambda
B -> switch ( E ) { 0 }
B -> if ( E ) S
O -> case constEnt : C D O
O -> default : C D O
O -> lambda
D -> break ;
D -> lambda
B -> let id T ;
T -> int
T -> boolean
T -> string
B -> S
F -> function id H ( A ) { C }
H -> T
H -> lambda
A -> T id K
A -> lambda
K -> , T id K
K -> lambda
C -> B C
C -> lambda
P -> B P
P -> F P
P -> lambda
PP -> P
}

```

## 2. Diseño del Generador de Código Intermedio

### 2.1. Traducción Dirigida Por la Sintaxis con Acciones Semánticas

Aprovechamos el diseño del Esquema de Traducción del procesador para añadir las nuevas acciones semánticas del Generador de Código Intermedio. A continuación se muestra una breve descripción de todas los métodos y funciones que aparecen en las acciones.

#### Métodos:

emite(operador, op1, op2, res): Escribe en el fichero de código intermedio la operación descrita por el cuarteto (algunos operandos pueden valer nulo).

buscaEtTS(idPos): Busca la etiqueta por la posición de la Tabla de Símbolos correspondiente del id que la representa.

nuevaEt(): Crea una nueva etiqueta en la Tabla de Símbolos.

tamRAcalculator(): Calcula cuánto ocupa el tamaño del Registro de Activación de una función en pila.

buscaTipoTS(idPos): Busca el tipo de un id según su posición en la Tabla de Símbolos correspondiente.

buscaLugarTS(idPos): Busca la posición en la Tabla de Símbolos correspondiente el id determinado.

#### Atributos de los No Terminales:

.evaluado: Determina si el case de un switch ya ha sido evaluado, para así no hacer las comprobaciones de los siguientes casos y los ejecute directamente si no hay break en dicho case.

.siguiente:

.pos: Determina cuál es la posición dentro de una tabla de símbolos

.lugar: Determina cuál es el lugar de la variable dentro de una tabla de símbolos

.break: Determina la etiqueta a la que se debe saltar dentro de un switch para saltar fuera del switch cuando se encuentra un break

.tipo: Determina el tipo del no terminal correspondiente

.etiq: Determina cual es la etiqueta que tiene una función o una string

.params: Es un Array que contiene el lugar de los parámetros que se van a pasar a la llamada a función siguiente

#### Bloque Axioma:

$$PP \rightarrow P \{\emptyset\}_1$$

$$P \rightarrow B P \{\emptyset\}_2$$

$$P \rightarrow F P \{\emptyset\}_3$$

$$P \rightarrow \lambda \{\emptyset\}_4$$

#### Bloque declaración de funciones:

$$F \rightarrow \text{function id H ( A ) } \{\text{emite(':', buscaEtTS(id.pos), NULL, NULL)}\}_{5,2} \\ \{ C \{\text{tamRAcalculator()}\}_{5,3} \} \{\text{emite('return', NULL, NULL, NULL)}\}_{5,4}$$

$H \rightarrow T \{\emptyset\}_6$

$H \rightarrow \lambda \{\emptyset\}_7$

$A \rightarrow T \text{ id } K \{\emptyset\}_8$

$A \rightarrow \lambda \{\emptyset\}_9$

$K \rightarrow , T \text{ id } K \{\emptyset\}_{10}$

$K \rightarrow \lambda \{\emptyset\}_{11}$

$C \rightarrow B C \{\emptyset\}_{12}$

$C \rightarrow \lambda \{\emptyset\}_{13}$

Bloque sentencias compuestas y declaración de variables:

$B \rightarrow \text{if } ( E ) \{ B.\text{siguiente} := \text{nuevaEt}(), \text{emite}(\text{'if'}, E.\text{lugar}, \text{NULL}, B.\text{siguiente}) \}_{14,1}$   
 $\quad S \{ \text{emite}(\text{'.'}, B.\text{siguiente}, \text{NULL}, \text{NULL}) \}_{14}$

$B \rightarrow \text{switch } ( E ) \{ O \} \{ \text{emite}(\text{'.'}, B.\text{break}, \text{NULL}, \text{NULL}) \}_{15,3}$

$O \rightarrow \text{case constEnt } \{ O.\text{break} := B.\text{break}, O.\text{evaluado} := B.\text{evaluado}$   
 $\quad O.\text{siguiente} := \text{nuevaEtq}(), O.\text{lugar} := E.\text{lugar}$   
 $\quad \text{emite}(\text{'if=='}, B.\text{evaluado}, 1, \text{sig\_inst} + 1)$   
 $\quad \text{emite}(\text{'if!='}, O.\text{lugar}, \text{constEnt}, O.\text{siguiente}) \}_{16,1}$   
 $\quad : C D \{ \text{emite}(\text{'.'}, 1, \text{NULL}, O.\text{evaluado}) \text{emite}(\text{'.'}, O.\text{siguiente}, \text{NULL}, \text{NULL}) \}_{16,3}$   
 $\quad O \{\emptyset\}_{16,2}$

$O \rightarrow \lambda \{\emptyset\}_{18}$

$D \rightarrow \text{break ; } \{ \text{emite}(\text{'goto'}, D.\text{break}, \text{NULL}, \text{NULL}) \}_{19}$

$D \rightarrow \lambda \{\emptyset\}_{20}$

$B \rightarrow \text{let id } T ; \{\emptyset\}_{21}$

$T \rightarrow \text{int } \{\emptyset\}_{22}$

$T \rightarrow \text{boolean } \{\emptyset\}_{23}$

$T \rightarrow \text{string } \{\emptyset\}_{24}$

$B \rightarrow S \{\emptyset\}_{25}$

Sentencias simples:

$S \rightarrow \text{id SS} \{ \text{SS.tipo} := \text{BuscaTipoTS}(\text{id.pos})$   
     $\text{SS.lugar} := \text{BuscaLugarTS}(\text{id.pos})$   
     $\text{if SS.tipo} == \text{funcion then SS.etiq} := \text{BuscaEtiqTS}(\text{id.pos}) \}_{26}$

$\text{SS} \rightarrow \% = E ; \{ \text{emite}(\text{'\%'}, E.\text{lugar}, \text{SS.lugar}, \text{SS.lugar}) \}_{27}$

$\text{SS} \rightarrow = E ; \{ \text{emite}(\text{':='} , E.\text{lugar}, \text{NULL}, \text{SS.lugar}) \}_{28}$

$\text{SS} \rightarrow ( L ) ; \{ \text{for (param in L.params)}$   
     $\text{emite}(\text{'param'}, \text{param}, \text{NULL}, \text{NULL})$   
     $\text{emite}(\text{'call'}, \text{SS.etiq}, \text{NULL}, \text{NULL}) \}_{29}$

$S \rightarrow \text{print R} ; \{ \text{emite}(\text{"print"}, R.\text{lugar}, \text{NULL}, \text{NULL}) \}_{30}$

$S \rightarrow \text{input id} ; \{ \text{emite}(\text{"input"}, \text{BuscaLugarTS}(\text{id}), \text{NULL}, \text{NULL}) \}_{31}$

$S \rightarrow \text{return X} ; \{ \text{emite}(\text{'return'}, X.\text{lugar}, \text{NULL}, \text{NULL}) \}_{32}$

$L \rightarrow E Q \{ L.\text{param} := E.\text{lugar} \oplus Q.\text{param} \}_{33}$

$L \rightarrow \lambda \{ \emptyset \}_{34}$

$Q \rightarrow , E Q1 \{ Q.\text{param} := E.\text{lugar} \oplus Q1.\text{param} \}_{35}$

$Q \rightarrow \lambda \{ \emptyset \}_{36}$

$X \rightarrow E \{ X.\text{lugar} := E.\text{lugar} \}_{37}$

$X \rightarrow \lambda \{ \emptyset \}_{38}$

Expresiones:

$E \rightarrow R \text{ RR} \{ \text{if RR.tipo} != \text{vacio}$   
     $\text{then E.lugar} := \text{nuevaTemp}(\text{logico})$   
     $\text{emite}(\text{'if=='}, R.\text{lugar}, \text{RR.lugar}, \text{goto } 2)$   
     $\text{emite}(\text{':='}, 0, \text{null}, E.\text{lugar})$   
     $\text{emite}(\text{'goto'}, \text{sigInst} + 1)$   
     $\text{emite}(\text{':='}, 1, \text{null}, E.\text{lugar})$   
     $\text{else then}$   
     $E.\text{lugar} := R.\text{lugar}$   
     $\text{emite}(\text{':='}, R.\text{lugar}, \text{null}, E.\text{lugar}) \}_{39}$

$\text{RR} \rightarrow == R \text{ RR} \{ \text{RR.lugar} := R.\text{lugar} \}_{40}$

$\text{RR} \rightarrow \lambda \{ \emptyset \}_{41}$

```

R → U UU {UU.lugar := nuevaTemp(constEnt)
            if UU1.tipo == vacio
            then emite(':', U.lugar, null, UU.lugar)
            else
            then emite('+', U.lugar, UU1.lugar, UU.lugar)}43

UU → + U UU {if (U.tipo = constEnt && UU2.tipo != tipo.error)
              then UU.tipo := tipo.ok
              else then UU.tipo := tipo.error
              error("error semantico: operando solo admite valores enteros");
              POP(3)}44

UU → λ {∅}45

U → ! V {U.lugar := nuevaTemp(boolean)
          emite('not', V.lugar, NULL, U.lugar)}46

U → V {U.lugar := V.lugar}47

V → id VV {if buscaTipoTS(id.pos) != function
            then V.lugar := buscaLugarTS(id.pos)
            else then
            VV.etiq := BuscaEtiqTS(id.pos)
            V.lugar = VV.lugar}48

V → ( E ) {V.lugar := E.lugar}49

V → constEnt {V.lugar := nuevaTemp(tipo.constEnt);
              emite(':', constEnt.valor, NULL, V.lugar)}50

V → cadena {V.lugar := nuevaTemp(tipo.cadena);
             emite(':', cadena.etiqueta, NULL, V.lugar)}51

VV → ( L ) {VV = nuevaTemp()
            for (param in L.params)
            emite('param', param, NULL, NULL)
            emite('call', VV.etiq, NULL, VV.lugar)}52

VV → λ {∅}53

```



### 3. Diseño del Generador de Código Final

#### 3.1. Plantilla de Traducción de Cuartetos

A continuación se muestran todos los posibles cuartetos generados por el Generador de Código Intermedio, y su correspondiente traducción a Código Objeto. Nótese que para cada operando que aparece en el cuarteto, se recogerá la dirección del mismo de una forma u otra (puede ser una constante, o estar en alguna tabla de símbolos...). También algunas traducciones pueden variar dependiendo del caso en el que estemos ejecutando (dentro de una función o desde main).

Emitiremos la traducción a ensamblador en el momento en el que se haga un emit del código intermedio.

- (+, op1, op2, res)

```
ADD op1, op2
MOVE .A, res
```

- (:=, op1, NULL, res)

```
MOVE op1, res
```

- (;, et01, null, null)

```
et01: NOP
```

- (% , op1, op2, res)

```
MOD op1, op2
MOVE .A, res
```

- (print, op1, null, null)

En el caso de que sea una string:

```
ADD #0, op1
WRSTR [.A]
```

Accedemos accedemos a lo que apunta el acumulador, ya que es una etiqueta a una string.

En el caso de que sea un entero:

```
WRINT op1
```

- (input, op1, null, null)

En el caso de que sea una string:

```
INSTR op1
```

En el caso de que sea un entero:

```
ININT op1
```

- (return, op1, null, null)
 

```

SUB #Tam_RA_X, #1
ADD .A, .IX
MOVE op1, [.A]
BR [.IX]

```
- (return, op1, null, null)
 

```

BR [.IX]

```
- (not, op1, null, res)
 

```

XOR op1, #1
MOVE .A, res

```
- (goto, null, null, sig\_int + 1)
 

```

BR $3

```
- (goto, et1, null, null)
 

```

BR /et1

```
- (if, op1, null, et01)
 

```

CMP op1, #1
BNZ /et01

```
- (if==, op1, op2, 2)
 

```

CMP op1, op2
BZ $5

```
- (if!=, op1, constEnt, et01)
 

```

CMP op1, constEnt
BNZ /et01

```
- (call, et1, null, res)
 

```

ADD #Tam_RA_X, .IX
MOVE #dir_ret_X, [.A]
MOVE .A, .IX
BR /etX
dir_ret_X: NOP
SUB #Tam_RA_X, #1
ADD .A, .IX
MOVE [.A], .R9
SUB .IX, #Tam_RA_X
MOVE .A, .IX
MOVE .R9, res

```

- (call, et1, null, null)

```

ADD #Tam_RA_X, .IX
MOVE #dir_ret_X, [.A]
MOVE .A, .IX
BR /etX
dir_ret_X: NOP
SUB .IX, #Tam_RA_X
MOVE .A, .IX

```

- (param, op1, null, null)

```

ADD #0, .IX
ADD #desp, .A
MOVE op1, [.A]

```

### 3.2. Tratamiento de las Strings

Las strings se tratan de manera que cada cadena de texto se almacena en el ensamblador con una etiqueta generada, esto es posible gracias a la instrucción DATA, en el caso de que la cadena este implícita en el código.

En el caso en el que la cadena es recibida por medio de input, se reservan 64 espacios de memoria con la instrucción RES 64 y una etiqueta para que se almacenene la string.

Todas las cadenas se inicializan a la string vacia, "".

## 4. Diseño del Entorno de Ejecución

### 4.1. Esquema de la memoria en ejecución

La asignación de la memoria se realiza como describe la siguiente figura:

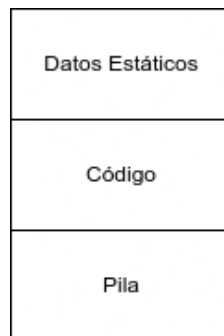


Figura 1: Estructuración de las zonas de memoria.

La primera parte de la memoria será la zona de Datos Estáticos, en donde se cargarán variables globales y datos temporales de la ejecución de main. La segunda zona son las instrucciones

en ensamblador. E inmediatamente debajo de ellas, está la zona de pila, que comienza justo después de la última instrucción emitida, y carga todos los Registros de Activación a partir de ellas. Esta última crece hacia posiciones crecientes de memoria.

Utilizamos los registros `.IX` para acceder a la zona de pila, y `.IY` para la zona de datos estáticos, ya que estos aceptan direccionamientos relativos.

## 4.2. Diseño del Registro de Activación

El Registro de Activación será apilado en la zona de memoria de la pila una consecutivamente de la otra. Está compuesto de los siguientes campos:

- Estado de la máquina: Contiene la información necesaria para restaurar la ejecución por donde se dejó antes de la llamada a la subrutina. En nuestro caso solo contendrá la dirección de retorno.
- Parámetros: Contiene los valores necesarios con los que se inicializan los parámetros que utiliza la función.
- Variables Locales: Todas las variables que se declaren en la función, se almacenarán sus valores en este campo.
- Datos Temporales: Aquí se almacenarán valores como constantes enteras, operaciones aritméticas o evaluaciones de condiciones booleanas o de parámetros a llamadas de otras funciones.
- Valor Devuelto: El llamado cargará en este campo el valor devuelto

EM
P
VL
DT
VD

Figura 2: Campos del Registro de Activación.

Los campos EM y P los carga el llamante, el resto los usa la propia función llamada. Y el valor devuelto se encarga de ubicarlo el llamado, para que lo pueda recoger el llamante posteriormente. En la práctica, los campos de variables locales y datos temporales se entremezclan entre ellos, ya que la función los utiliza de forma indiscriminada.

## A. Casos de Prueba

### A.1. Prueba Funcional 1

Código fuente:

```
1 let a int;
2 let b int ;
3 let bbb boolean ;
4 a = 3;
5 b=a ;
6 let c boolean ;
7 c = a == b;
8 if (c) b %= 1;
9 if (b == a) b = 44;
10 a = a + b;
11 print a ;
12 print b;
```

Fichero de lenguaje intermedio generado:

```
1 := 3 null 0.4
2 := 0.4 null 0.1
3 := 0.1 null 0.2
4 if== 0.1 0.2 goto 2
5 := 0 null 0.6
6 goto null null 1
7 := 1 null 0.6
8 := 0.6 null 0.5
9 if 0.5 null et01
10 := 1 null 0.7
11 % 0.7 0.2 0.2
12 : et01 null null
13 if== 0.2 0.1 goto 2
14 := 0 null 0.8
15 goto null null 1
16 := 1 null 0.8
17 if 0.8 null et02
18 := 44 null 0.9
19 := 0.9 null 0.2
20 : et02 null null
21 := 0.2 null 0.10
22 + 0.1 0.10 0.11
23 := 0.11 null 0.1
24 print 0.1 null null
25 print 0.2 null null
```

Fichero de lenguaje objeto generado:

```
1 MOVE #inicio_estaticas, .IY
2 MOVE #inicio_pila, .IX
3 BR /main
4
5 main:
6     MOVE #3, #4[.IY]
7     MOVE #4[.IY], #1[.IY]
8     MOVE #1[.IY], #2[.IY]
9     CMP #1[.IY], #2[.IY]
10    BZ 5MOVE0,6[.IY]BR3
11    MOVE #1, #6[.IY]
12    MOVE #6[.IY], #5[.IY]
13    CMP #5[.IY], #1
```

```

14     BNZ /et01
15     MOVE #1, #7[.IY]
16     MOD #2[.IY], #7[.IY]
17     MOVE .A, #2[.IY]
18 et01: NOP
19     CMP #2[.IY], #1[.IY]
20     BZ 5MOVE0,8[.IY]BR3
21     MOVE #1, #8[.IY]
22     CMP #8[.IY], #1
23     BNZ /et02
24     MOVE #44, #9[.IY]
25     MOVE #9[.IY], #2[.IY]
26 et02: NOP
27     MOVE #2[.IY], #10[.IY]
28     ADD #1[.IY], #10[.IY]
29     MOVE .A, #11[.IY]
30     MOVE #11[.IY], #1[.IY]
31     WRINT #1[.IY]
32     WRINT #2[.IY]
33     HALT
34
35 inicio_estaticas: RES 11
36
37
38 inicio_pila: NOP
39
40 END
41 ; una obra de diriG studios(c) 2023.

```

Fichero de Tabla de simbolos generado:

```

1 #0:
2 * LEXEMA : 'a'
3   ATRIBUTOS :
4     +tipo : 'constEnt'
5     +despl : 0
6 * LEXEMA : 'b'
7   ATRIBUTOS :
8     +tipo : 'constEnt'
9     +despl : 1
10 * LEXEMA : 'bbb'
11  ATRIBUTOS :
12    +tipo : 'booleanR'
13    +despl : 2
14 * LEXEMA : 'temp31'
15  ATRIBUTOS :
16    +tipo : 'constEnt'
17    +despl : 3
18 * LEXEMA : 'c'
19  ATRIBUTOS :
20    +tipo : 'booleanR'
21    +despl : 4
22 * LEXEMA : 'temp51'
23  ATRIBUTOS :
24    +tipo : 'booleanR'
25    +despl : 5
26 * LEXEMA : 'temp61'
27  ATRIBUTOS :
28    +tipo : 'constEnt'
29    +despl : 6
30 * LEXEMA : 'temp71'
31  ATRIBUTOS :
32    +tipo : 'booleanR'

```

```

33     +despl : 7
34 * LEXEMA : 'temp81'
35   ATRIBUTOS :
36     +tipo : 'constEnt'
37     +despl : 8
38 * LEXEMA : 'temp91'
39   ATRIBUTOS :
40     +tipo : 'constEnt'
41     +despl : 9
42 * LEXEMA : 'temp101'
43   ATRIBUTOS :
44     +tipo : 'constEnt'
45     +despl : 10

```

En este primer ejemplo probamos la declaración de variables, la asignación, con y sin resto, las comparaciones y los if simples, así como el print.

## A.2. Prueba Funcional 2

Código fuente:

```

1 let a    int    ;
2 let b    int    ;
3 let number int ;
4 print "Introduce el primer operando" ;
5 input a;
6 print "Introduce el segundo operando";input
7   b;
8 function operacion int(int num2,int num1)
9 {
10   let res int;
11   res=num1+num2;
12   return ((res));
13 }
14 number = 0;
15 print operacion(b,a);

```

Fichero de lenguaje intermedio generado:

```

1 := "Introduce el primer operando" null 0.4
2 print 0.4 null null
3 input 0.1 null null
4 := "Introduce el segundo operando" null 0.5
5 print 0.5 null null
6 input 0.2 null null
7 : Etoperacion null null
8 := 1.1 null 1.4
9 + 1.2 1.4 1.5
10 := 1.5 null 1.3
11 return 1.3 null null
12 return null null null
13 := 0 null 0.7
14 := 0.7 null 0.3
15 param 0.2 null null
16 param 0.1 null null
17 call Etoperacion null 0.8
18 print 0.8 null null

```

Fichero de lenguaje objeto generado:

```

1 MOVE #inicio_estaticas, .IY
2 MOVE #inicio_pila, .IX

```

```

3 BR /main
4
5 main:
6     MOVE #data01, #4[.IY]
7     ADD #0, #4[.IY]
8     WRSTR [.A]
9     ININT #1[.IY]
10    MOVE #data02, #5[.IY]
11    ADD #0, #5[.IY]
12    WRSTR [.A]
13    ININT #2[.IY]
14    MOVE #0, #7[.IY]
15    MOVE #7[.IY], #3[.IY]
16    ADD #0, .IX
17    ADD #1, .A
18    MOVE #2[.IY], [.A]
19    ADD #0, .IX
20    ADD #2, .A
21    MOVE #1[.IY], [.A]
22    MOVE #dir_ret_1, [.IX]
23    BR /Etoperacion
24 dir_ret_1: NOP
25    MOVE [.A], #8[.IY]
26    MOVE #inicio_pila, .IX
27    WRINT #8[.IY]
28    HALT
29
30 Etoperacion: NOP
31    MOVE #1[.IX], #4[.IX]
32    ADD #2[.IX], #4[.IX]
33    MOVE .A, #5[.IX]
34    MOVE #5[.IX], #3[.IX]
35    SUB #Tam_RA_Etoperacion, #1
36    ADD .A, .IX;
37    MOVE #3[.IX], [.A]
38    BR [.IX]
39
40    BR [.IX]
41
42 Tam_RA_Etoperacion: EQU 7
43 inicio_estaticas: RES 8
44
45 data01: DATA "Introduce el primer operando "
46 data02: DATA "Introduce el segundo operando "
47
48 inicio_pila: NOP
49
50 END
51 ; una obra de diriG studios(c) 2023.

```

Fichero de Tabla de simbolos generado:

```

1 #1:
2 * LEXEMA : 'num2'
3   ATRIBUTOS :
4     +tipo : 'constEnt'
5     +despl : 0
6 * LEXEMA : 'num1'
7   ATRIBUTOS :
8     +tipo : 'constEnt'
9     +despl : 1
10 * LEXEMA : 'res'
11   ATRIBUTOS :

```



```

12     +tipo : 'constEnt'
13     +despl : 2
14 * LEXEMA : 'temp31'
15   ATRIBUTOS :
16     +tipo : 'constEnt'
17     +despl : 3
18 * LEXEMA : 'temp41'
19   ATRIBUTOS :
20     +tipo : 'constEnt'
21     +despl : 4
22 #0:
23 * LEXEMA : 'a'
24   ATRIBUTOS :
25     +tipo : 'constEnt'
26     +despl : 0
27 * LEXEMA : 'b'
28   ATRIBUTOS :
29     +tipo : 'constEnt'
30     +despl : 1
31 * LEXEMA : 'number'
32   ATRIBUTOS :
33     +tipo : 'constEnt'
34     +despl : 2
35 * LEXEMA : 'temp31'
36   ATRIBUTOS :
37     +tipo : 'cadena'
38     +despl : 3
39 * LEXEMA : 'temp41'
40   ATRIBUTOS :
41     +tipo : 'cadena'
42     +despl : 4
43 * LEXEMA : 'operacion'
44   ATRIBUTOS :
45     +tipo : 'funcion'
46     +numParam : 2
47     +TipoParam1 : 'constEnt'
48     +ModoParam1 : 1
49     +TipoParam2 : 'constEnt'
50     +ModoParam2 : 1
51     +TipoRetorno : 'constEnt'
52     +EtiqFuncion : 'Etoperacion01'
53 * LEXEMA : 'temp61'
54   ATRIBUTOS :
55     +tipo : 'constEnt'
56     +despl : 5
57 * LEXEMA : 'temp71'
58   ATRIBUTOS :
59     +tipo : 'constEnt'
60     +despl : 6

```

En este segundo ejemplo probamos la declaración de variables, el print de cadenas, las llamadas a funciones que retornan valor, y el uso de print de un entero con llamada a una función.

### A.3. Prueba Funcional 3

Código fuente:

```

1 let texto string;
2 function alert (string m_s_g)
3 {
4   print "Mensaje introducido:";

```

```

5   print m_s_g;
6   }
7   function pideTexto ()
8   {
9       print "Introduce un texto corto";
10      input texto;
11  }
12  pideTexto();
13  let  textoAux  string;textoAux = texto;
14  alert
15  (
16      textoAux
17  )
18  ;

```

Fichero de lenguaje intermedio generado:

```

1  := "" null 0.1
2  : Etalert null null
3  := "Mensaje introducido:" null 1.2
4  print 1.2 null null
5  print 1.1 null null
6  return null null null
7  : EtpideTexto null null
8  := "Introduce un texto corto" null 2.1
9  print 2.1 null null
10 input 0.1 null null
11 return null null null
12 call EtpideTexto null null
13 := "" null 0.4
14 := 0.1 null 0.4
15 param 0.4 null null
16 call Etalert null null

```

Fichero de lenguaje objeto generado:

```

1  MOVE #inicio_estaticas, .IY
2  MOVE #inicio_pila, .IX
3  BR /main
4
5  main:
6      MOVE #data01, #1[.IY]
7      MOVE #dir_ret_1, [.IX]
8      BR /EtpideTexto
9  dir_ret_1:  NOP
10     MOVE #inicio_pila, .IX
11     MOVE #data05, #4[.IY]
12     MOVE #1[.IY], #4[.IY]
13     ADD #0, .IX
14     ADD #1, .A
15     MOVE #4[.IY], [.A]
16     MOVE #dir_ret_2, [.IX]
17     BR /Etalert
18 dir_ret_2:  NOP
19     MOVE #inicio_pila, .IX
20     HALT
21
22 Etalert:  NOP
23     MOVE #data02, #2[.IX]
24     ADD #0, #2[.IX]
25     WRSTR [.A]
26     ADD #0, #1[.IX]
27     WRSTR [.A]

```

```

28      BR [.IX]
29
30
31 EtpideTexto:  NOP
32      MOVE #data03, #1[.IX]
33      ADD #0, #1[.IX]
34      WRSTR [.A]
35      INSTR /data04
36      MOVE #data04, #1[.IY]
37
38      BR [.IX]
39
40 Tam_RA_Etalert: EQU 4
41 Tam_RA_EtpideTexto: EQU 3
42 inicio_estaticas: RES 4
43
44 data01: DATA " "
45 data02: DATA "Mensaje introducido: "
46 data03: DATA "Introduce un texto corto "
47 data04: RES 64
48 data05: DATA " "
49
50 inicio_pila: NOP
51
52 END
53 ; una obra de diriG studios(c) 2023.

```

Fichero de Tabla de simbolos generado:

```

1 #1:
2 * LEXEMA : 'm_s_g'
3   ATRIBUTOS :
4     +tipo : 'cadena'
5     +despl : 0
6 * LEXEMA : 'temp11'
7   ATRIBUTOS :
8     +tipo : 'cadena'
9     +despl : 1
10 #2:
11 * LEXEMA : 'temp01'
12   ATRIBUTOS :
13     +tipo : 'cadena'
14     +despl : 0
15 #0:
16 * LEXEMA : 'texto'
17   ATRIBUTOS :
18     +tipo : 'cadena'
19     +despl : 0
20 * LEXEMA : 'alert'
21   ATRIBUTOS :
22     +tipo : 'funcion'
23     +numParam : 1
24     +TipoParam1 : 'cadena'
25     +ModoParam1 : 1
26     +TipoRetorno : 'vacio'
27     +EtiqFuncion : 'Etalert01'
28 * LEXEMA : 'pideTexto'
29   ATRIBUTOS :
30     +tipo : 'funcion'
31     +numParam : 0
32     +TipoRetorno : 'vacio'
33     +EtiqFuncion : 'EtpideTexto01'
34 * LEXEMA : 'textoAux'

```

```

35  ATRIBUTOS :
36      +tipo : 'cadena'
37      +despl : 1

```

En este ejemplo realizamos varias pruebas de llamada a funciones, con input's de string y print de ellos, además de probar el funcionamiento de tanto variables locales como globales.

#### A.4. Prueba Funcional 4

Código fuente:

```

1  function recursiva int (int a)
2  {
3      switch (a)
4      {
5          case 1:
6              recursiva(a + 1);
7              break;
8          case 2:
9              print("segundo case");
10         case 3:
11             print("tercer case");
12             break;
13     }
14     return 14;
15 }
16
17 let entero int;
18 entero = 14;
19 if (recursiva(1) == entero)
20     print(entero);

```

Fichero de lenguaje intermedio generado:

```

1  : Etrecurativa null null
2  if== 1.2 1 2
3  if!= 1.1 1 et02
4  := 1 null 1.3
5  := 1.3 null 1.4
6  + 1.1 1.4 1.5
7  param 1.5 null null
8  call Etrecurativa null null
9  goto et01 null null
10 := 1 null 1.2
11 : et02 null null
12 if== 1.2 1 2
13 if!= 1.1 2 et03
14 := "segundo case" null 1.6
15 print 1.6 null null
16 := 1 null 1.2
17 : et03 null null
18 if== 1.2 1 2
19 if!= 1.1 3 et04
20 := "tercer case" null 1.7
21 print 1.7 null null
22 goto et01 null null
23 := 1 null 1.2
24 : et04 null null
25 : et01 null null
26 := 14 null 1.8
27 return 1.8 null null
28 return null null null

```

```

29 := 14 null 0.3
30 := 0.3 null 0.2
31 := 1 null 0.4
32 param 0.4 null null
33 call Etrecursiva null 0.5
34 if== 0.5 0.2 goto 2
35 := 0 null 0.6
36 goto null null 1
37 := 1 null 0.6
38 if 0.6 null et05
39 print 0.2 null null
40 : et05 null null

```

Fichero de lenguaje objeto generado:

```

1 MOVE #inicio_estaticas, .IY
2 MOVE #inicio_pila, .IX
3 BR /main
4
5 main:
6     MOVE #14, #3[.IY]
7     MOVE #3[.IY], #2[.IY]
8     MOVE #1, #4[.IY]
9     ADD #0, .IX
10    ADD #1, .A
11    MOVE #4[.IY], [.A]
12    MOVE #dir_ret_2, [.IX]
13    BR /Etrecursiva
14 dir_ret_2: NOP
15    MOVE [.A], #5[.IY]
16    MOVE #inicio_pila, .IX
17    CMP #5[.IY], #2[.IY]
18    BZ $5
19    MOVE #0, #6[.IY]
20    BR $3
21    MOVE #1, #6[.IY]
22    CMP #6[.IY], #1
23    BNZ /et05
24    WRINT #2[.IY]
25 et05: NOP
26    HALT
27
28 Etrecursiva: NOP
29    CMP #2[.IX], #1
30    BZ $5
31    CMP #1[.IX], #1
32    BNZ /et02
33    MOVE #1, #3[.IX]
34    MOVE #3[.IX], #4[.IX]
35    ADD #1[.IX], #4[.IX]
36    MOVE .A, #5[.IX]
37    ADD #Tam_RA_Etrecursiva, .IX
38    ADD #1, .A
39    MOVE #5[.IX], [.A]
40    ADD #Tam_RA_Etrecursiva, .IX
41    MOVE #dir_ret_1, [.A]
42    MOVE .A, .IX
43    BR /Etrecursiva
44 dir_ret_1: NOP
45    SUB .IX, #Tam_RA_Etrecursiva
46    MOVE .A, .IX
47    BR /et01
48    MOVE #1, #2[.IX]

```

```

49 et02: NOP
50     CMP #2[.IX],#1
51     BZ $5
52     CMP #1[.IX], #2
53     BNZ /et03
54     MOVE #data01, #6[.IX]
55     ADD #0, #6[.IX]
56     WRSTR [.A]
57     MOVE #1, #2[.IX]
58 et03: NOP
59     CMP #2[.IX],#1
60     BZ $5
61     CMP #1[.IX], #3
62     BNZ /et04
63     MOVE #data02, #7[.IX]
64     ADD #0, #7[.IX]
65     WRSTR [.A]
66     BR /et01
67     MOVE #1, #2[.IX]
68 et04: NOP
69 et01: NOP
70     MOVE #14, #8[.IX]
71     SUB #Tam_RA_Etrecursiva, #1
72     ADD .A, .IX;
73     MOVE #8[.IX], [.A]
74     BR [.IX]
75
76     BR [.IX]
77
78 Tam_RA_Etrecursiva: EQU 10
79 inicio_estaticas: RES 6
80
81 data01: DATA "segundo case "
82 data02: DATA "tercer case "
83
84 inicio_pila: NOP
85
86 END
87 ; una obra de diriG studios(c) 2023.

```

Fichero de Tabla de simbolos generado:

```

1 #1:
2 * LEXEMA : 'a'
3   ATRIBUTOS :
4     +tipo : 'constEnt'
5     +despl : 0
6 * LEXEMA : 'temp11'
7   ATRIBUTOS :
8     +tipo : 'constEnt'
9     +despl : 1
10 * LEXEMA : 'temp21'
11   ATRIBUTOS :
12     +tipo : 'constEnt'
13     +despl : 2
14 * LEXEMA : 'temp31'
15   ATRIBUTOS :
16     +tipo : 'constEnt'
17     +despl : 3
18 * LEXEMA : 'temp41'
19   ATRIBUTOS :
20     +tipo : 'constEnt'
21     +despl : 4

```

```

22 * LEXEMA : 'temp51'
23   ATRIBUTOS :
24     +tipo : 'cadena'
25     +despl : 5
26 * LEXEMA : 'temp61'
27   ATRIBUTOS :
28     +tipo : 'cadena'
29     +despl : 6
30 * LEXEMA : 'temp71'
31   ATRIBUTOS :
32     +tipo : 'constEnt'
33     +despl : 7
34 #0:
35 * LEXEMA : 'recursiva'
36   ATRIBUTOS :
37     +tipo : 'funcion'
38     +numParam : 1
39     +TipoParam1 : 'constEnt'
40     +ModoParam1 : 1
41     +TipoRetorno : 'constEnt'
42     +EtiqFuncion : 'Etrecurativa01'
43 * LEXEMA : 'entero'
44   ATRIBUTOS :
45     +tipo : 'constEnt'
46     +despl : 0
47 * LEXEMA : 'temp21'
48   ATRIBUTOS :
49     +tipo : 'constEnt'
50     +despl : 1
51 * LEXEMA : 'temp31'
52   ATRIBUTOS :
53     +tipo : 'constEnt'
54     +despl : 2
55 * LEXEMA : 'temp41'
56   ATRIBUTOS :
57     +tipo : 'constEnt'
58     +despl : 3
59 * LEXEMA : 'temp51'
60   ATRIBUTOS :
61     +tipo : 'booleanR'
62     +despl : 4

```

En este ejemplo realizamos llamadas recursivas a una función para probar el correcto funcionamiento de la recursividad.

## A.5. Prueba Funcional 5

Código fuente:

```

1 let booleano boolean;
2 function bisiestro boolean (int a)
3 { let bis string;
4   print "Es bisiestro?";
5   input bis;
6   return ((a + 4 == 0));
7 }
8 function dias int (int m, int a)
9 {
10  switch (m)
11  {
12    case 1: case 3: case 5: case 7: case 8: case 10: case 12:

```

```

13     return 31; break;
14     case 4: case 6: case 9: case 11:
15         return 30;
16     case 2: if (bisiesto (a)) return 29;
17         return(28);
18
19 }
20 }
21 function esFechaCorrecta boolean (int d, int m, int a)
22 {
23     return (d == dias (m, a));
24 }
25 function demo ()
26 {
27
28     if (esFechaCorrecta(31, 08, 2022)) print 9999;
29     print("     se ha comprobado         la fecha");
30     return;
31 }
32 let abc int;
33 demo();

```

Fichero de lenguaje intermedio generado:

```

1 : Etbisiesto null null
2 := " " null 1.2
3 := "Es bisiesto?" null 1.3
4 print 1.3 null null
5 input 1.2 null null
6 := 4 null 1.4
7 := 1.4 null 1.5
8 + 1.1 1.5 1.6
9 := 0 null 1.7
10 if== 1.6 1.7 goto 2
11 := 0 null 1.8
12 goto null null 1
13 := 1 null 1.8
14 return 1.8 null null
15 return null null null
16 : Etdias null null
17 if== 2.3 1 2
18 if!= 2.1 1 et02
19 := 1 null 2.3
20 : et02 null null
21 if== 2.3 1 2
22 if!= 2.1 3 et03
23 := 1 null 2.3
24 : et03 null null
25 if== 2.3 1 2
26 if!= 2.1 5 et04
27 := 1 null 2.3
28 : et04 null null
29 if== 2.3 1 2
30 if!= 2.1 7 et05
31 := 1 null 2.3
32 : et05 null null
33 if== 2.3 1 2
34 if!= 2.1 8 et06
35 := 1 null 2.3
36 : et06 null null
37 if== 2.3 1 2
38 if!= 2.1 10 et07
39 := 1 null 2.3

```



```

40 : et07 null null
41 if== 2.3 1 2
42 if!= 2.1 12 et08
43 := 31 null 2.4
44 return 2.4 null null
45 goto et01 null null
46 := 1 null 2.3
47 : et08 null null
48 if== 2.3 1 2
49 if!= 2.1 4 et09
50 := 1 null 2.3
51 : et09 null null
52 if== 2.3 1 2
53 if!= 2.1 6 et010
54 := 1 null 2.3
55 : et010 null null
56 if== 2.3 1 2
57 if!= 2.1 9 et011
58 := 1 null 2.3
59 : et011 null null
60 if== 2.3 1 2
61 if!= 2.1 11 et012
62 := 30 null 2.5
63 return 2.5 null null
64 := 1 null 2.3
65 : et012 null null
66 if== 2.3 1 2
67 if!= 2.1 2 et013
68 param 2.2 null null
69 call Etbisiesto null 2.6
70 if 2.6 null et014
71 := 29 null 2.7
72 return 2.7 null null
73 : et014 null null
74 := 28 null 2.8
75 return 2.8 null null
76 := 1 null 2.3
77 : et013 null null
78 : et01 null null
79 return null null null
80 : EtesFechaCorrecta null null
81 param 3.2 null null
82 param 3.3 null null
83 call Etdias null 3.4
84 if== 3.1 3.4 goto 2
85 := 0 null 3.5
86 goto null null 1
87 := 1 null 3.5
88 return 3.5 null null
89 return null null null
90 : Etdemo null null
91 := 31 null 4.1
92 := 8 null 4.2
93 := 2022 null 4.3
94 param 4.1 null null
95 param 4.2 null null
96 param 4.3 null null
97 call EtesFechaCorrecta null 4.4
98 if 4.4 null et015
99 := 9999 null 4.5
100 print 4.5 null null
101 : et015 null null

```

```

102 := " se ha comprobado la fecha" null 4.6
103 print 4.6 null null
104 return null null null
105 return null null null
106 call Etdemo null null

```

Fichero de lenguaje objeto generado:

```

1 MOVE #inicio_estaticas, .IY
2 MOVE #inicio_pila, .IX
3 BR /main
4
5 main:
6     MOVE #dir_ret_4, [.IX]
7     BR /Etdemo
8 dir_ret_4: NOP
9     MOVE #inicio_pila, .IX
10    HALT
11
12 Etbisiesto: NOP
13    MOVE #data01, #2[.IX]
14    MOVE #data02, #3[.IX]
15    ADD #0, #3[.IX]
16    WRSTR [.A]
17    INSTR /data03
18    MOVE #data03, #2[.IX]
19    MOVE #4, #4[.IX]
20    MOVE #4[.IX], #5[.IX]
21    ADD #1[.IX], #5[.IX]
22    MOVE .A, #6[.IX]
23    MOVE #0, #7[.IX]
24    CMP #6[.IX], #7[.IX]
25    BZ $5
26    MOVE #0, #8[.IX]
27    BR $3
28    MOVE #1, #8[.IX]
29    SUB #Tam_RA_Etbisiesto, #1
30    ADD .A, .IX;
31    MOVE #8[.IX], [.A]
32    BR [.IX]
33
34    BR [.IX]
35
36 EtesFechaCorrecta: NOP
37    ADD #Tam_RA_EtesFechaCorrecta, .IX
38    ADD #1, .A
39    MOVE #2[.IX], [.A]
40    ADD #Tam_RA_EtesFechaCorrecta, .IX
41    ADD #2, .A
42    MOVE #3[.IX], [.A]
43    ADD #Tam_RA_EtesFechaCorrecta, .IX
44    MOVE #dir_ret_2, [.A]
45    MOVE .A, .IX
46    BR /Etdias
47 dir_ret_2: NOP
48    SUB #Tam_RA_Etdias, #1
49    ADD .A, .IX
50    MOVE [.A], .R9
51    SUB .IX, #Tam_RA_EtesFechaCorrecta
52    MOVE .A, .IX
53    MOVE .R9, #4[.IX]
54    CMP #1[.IX], #4[.IX]
55    BZ $5

```

```

56     MOVE #0, #5[.IX]
57     BR $3
58     MOVE #1, #5[.IX]
59     SUB #Tam_RA_EtesFechaCorrecta, #1
60     ADD .A, .IX;
61     MOVE #5[.IX], [.A]
62     BR [.IX]
63
64     BR [.IX]
65
66 Etdias: NOP
67     CMP #3[.IX], #1
68     BZ $5
69     CMP #1[.IX], #1
70     BNZ /et02
71     MOVE #1, #3[.IX]
72 et02: NOP
73     CMP #3[.IX], #1
74     BZ $5
75     CMP #1[.IX], #3
76     BNZ /et03
77     MOVE #1, #3[.IX]
78 et03: NOP
79     CMP #3[.IX], #1
80     BZ $5
81     CMP #1[.IX], #5
82     BNZ /et04
83     MOVE #1, #3[.IX]
84 et04: NOP
85     CMP #3[.IX], #1
86     BZ $5
87     CMP #1[.IX], #7
88     BNZ /et05
89     MOVE #1, #3[.IX]
90 et05: NOP
91     CMP #3[.IX], #1
92     BZ $5
93     CMP #1[.IX], #8
94     BNZ /et06
95     MOVE #1, #3[.IX]
96 et06: NOP
97     CMP #3[.IX], #1
98     BZ $5
99     CMP #1[.IX], #10
100    BNZ /et07
101    MOVE #1, #3[.IX]
102 et07: NOP
103    CMP #3[.IX], #1
104    BZ $5
105    CMP #1[.IX], #12
106    BNZ /et08
107    MOVE #31, #4[.IX]
108    SUB #Tam_RA_Etdias, #1
109    ADD .A, .IX;
110    MOVE #4[.IX], [.A]
111    BR [.IX]
112    BR /et01
113    MOVE #1, #3[.IX]
114 et08: NOP
115    CMP #3[.IX], #1
116    BZ $5
117    CMP #1[.IX], #4

```

```

118     BNZ /et09
119     MOVE #1, #3[.IX]
120 et09: NOP
121     CMP #3[.IX],#1
122     BZ $5
123     CMP #1[.IX], #6
124     BNZ /et010
125     MOVE #1, #3[.IX]
126 et010: NOP
127     CMP #3[.IX],#1
128     BZ $5
129     CMP #1[.IX], #9
130     BNZ /et011
131     MOVE #1, #3[.IX]
132 et011: NOP
133     CMP #3[.IX],#1
134     BZ $5
135     CMP #1[.IX], #11
136     BNZ /et012
137     MOVE #30, #5[.IX]
138     SUB #Tam_RA_Etdias, #1
139     ADD .A, .IX;
140     MOVE #5[.IX], [.A]
141     BR [.IX]
142     MOVE #1, #3[.IX]
143 et012: NOP
144     CMP #3[.IX],#1
145     BZ $5
146     CMP #1[.IX], #2
147     BNZ /et013
148     ADD #Tam_RA_Etdias, .IX
149     ADD #1, .A
150     MOVE #2[.IX], [.A]
151     ADD #Tam_RA_Etdias, .IX
152     MOVE #dir_ret_1, [.A]
153     MOVE .A, .IX
154     BR /Etbisiesto
155 dir_ret_1: NOP
156     SUB #Tam_RA_Etbisiesto, #1
157     ADD .A, .IX
158     MOVE [.A], .R9
159     SUB .IX, #Tam_RA_Etdias
160     MOVE .A, .IX
161     MOVE .R9, #6[.IX]
162     CMP #6[.IX], #1
163     BNZ /et014
164     MOVE #29, #7[.IX]
165     SUB #Tam_RA_Etdias, #1
166     ADD .A, .IX;
167     MOVE #7[.IX], [.A]
168     BR [.IX]
169 et014: NOP
170     MOVE #28, #8[.IX]
171     SUB #Tam_RA_Etdias, #1
172     ADD .A, .IX;
173     MOVE #8[.IX], [.A]
174     BR [.IX]
175     MOVE #1, #3[.IX]
176 et013: NOP
177 et01: NOP
178
179     BR [.IX]

```

```

180
181 Etdemo: NOP
182     MOVE #31, #1[.IX]
183     MOVE #8, #2[.IX]
184     MOVE #2022, #3[.IX]
185     ADD #Tam_RA_Etdemo, .IX
186     ADD #1, .A
187     MOVE #1[.IX], [.A]
188     ADD #Tam_RA_Etdemo, .IX
189     ADD #2, .A
190     MOVE #2[.IX], [.A]
191     ADD #Tam_RA_Etdemo, .IX
192     ADD #3, .A
193     MOVE #3[.IX], [.A]
194     ADD #Tam_RA_Etdemo, .IX
195     MOVE #dir_ret_3, [.A]
196     MOVE .A, .IX
197     BR /EtesFechaCorrecta
198 dir_ret_3: NOP
199     SUB #Tam_RA_EtesFechaCorrecta, #1
200     ADD .A, .IX
201     MOVE [.A], .R9
202     SUB .IX, #Tam_RA_Etdemo
203     MOVE .A, .IX
204     MOVE .R9, #4[.IX]
205     CMP #4[.IX], #1
206     BNZ /et015
207     MOVE #9999, #5[.IX]
208     WRINT #5[.IX]
209 et015: NOP
210     MOVE #data04, #6[.IX]
211     ADD #0, #6[.IX]
212     WRSTR [.A]
213     BR [.IX]
214
215     BR [.IX]
216
217 Tam_RA_Etbisiesto: EQU 10
218 Tam_RA_Etdias: EQU 10
219 Tam_RA_EtesFechaCorrecta: EQU 7
220 Tam_RA_Etdemo: EQU 8
221 inicio_estaticas: RES 6
222
223 data01: DATA " "
224 data02: DATA "Es bisiesto? "
225 data03: RES 64
226 data04: DATA " se ha comprobado la fecha "
227
228 inicio_pila: NOP
229
230 END
231 ; una obra de diriG studios(c) 2023.

```

Fichero de Tabla de simbolos generado:

```

1 #1:
2 * LEXEMA : 'a'
3   ATRIBUTOS :
4     +tipo : 'constEnt'
5     +despl : 0
6 * LEXEMA : 'bis'
7   ATRIBUTOS :
8     +tipo : 'cadena'

```

```

9      +despl : 1
10 * LEXEMA : 'temp21'
11   ATRIBUTOS :
12     +tipo : 'cadena'
13     +despl : 2
14 * LEXEMA : 'temp31'
15   ATRIBUTOS :
16     +tipo : 'constEnt'
17     +despl : 3
18 * LEXEMA : 'temp41'
19   ATRIBUTOS :
20     +tipo : 'constEnt'
21     +despl : 4
22 * LEXEMA : 'temp51'
23   ATRIBUTOS :
24     +tipo : 'constEnt'
25     +despl : 5
26 * LEXEMA : 'temp61'
27   ATRIBUTOS :
28     +tipo : 'constEnt'
29     +despl : 6
30 * LEXEMA : 'temp71'
31   ATRIBUTOS :
32     +tipo : 'booleanR'
33     +despl : 7
34 #2:
35 * LEXEMA : 'm'
36   ATRIBUTOS :
37     +tipo : 'constEnt'
38     +despl : 0
39 * LEXEMA : 'a'
40   ATRIBUTOS :
41     +tipo : 'constEnt'
42     +despl : 1
43 * LEXEMA : 'temp21'
44   ATRIBUTOS :
45     +tipo : 'constEnt'
46     +despl : 2
47 * LEXEMA : 'temp31'
48   ATRIBUTOS :
49     +tipo : 'constEnt'
50     +despl : 3
51 * LEXEMA : 'temp41'
52   ATRIBUTOS :
53     +tipo : 'constEnt'
54     +despl : 4
55 * LEXEMA : 'temp51'
56   ATRIBUTOS :
57     +tipo : 'booleanR'
58     +despl : 5
59 * LEXEMA : 'temp61'
60   ATRIBUTOS :
61     +tipo : 'constEnt'
62     +despl : 6
63 * LEXEMA : 'temp71'
64   ATRIBUTOS :
65     +tipo : 'constEnt'
66     +despl : 7
67 #3:
68 * LEXEMA : 'd'
69   ATRIBUTOS :
70     +tipo : 'constEnt'

```

```

71     +despl : 0
72 * LEXEMA : 'm'
73   ATRIBUTOS :
74     +tipo : 'constEnt'
75     +despl : 1
76 * LEXEMA : 'a'
77   ATRIBUTOS :
78     +tipo : 'constEnt'
79     +despl : 2
80 * LEXEMA : 'temp31'
81   ATRIBUTOS :
82     +tipo : 'constEnt'
83     +despl : 3
84 * LEXEMA : 'temp41'
85   ATRIBUTOS :
86     +tipo : 'booleanR'
87     +despl : 4
88 #4:
89 * LEXEMA : 'temp01'
90   ATRIBUTOS :
91     +tipo : 'constEnt'
92     +despl : 0
93 * LEXEMA : 'temp11'
94   ATRIBUTOS :
95     +tipo : 'constEnt'
96     +despl : 1
97 * LEXEMA : 'temp21'
98   ATRIBUTOS :
99     +tipo : 'constEnt'
100    +despl : 2
101 * LEXEMA : 'temp31'
102   ATRIBUTOS :
103     +tipo : 'booleanR'
104     +despl : 3
105 * LEXEMA : 'temp41'
106   ATRIBUTOS :
107     +tipo : 'constEnt'
108     +despl : 4
109 * LEXEMA : 'temp51'
110   ATRIBUTOS :
111     +tipo : 'cadena'
112     +despl : 5
113 #0:
114 * LEXEMA : 'booleano'
115   ATRIBUTOS :
116     +tipo : 'booleanR'
117     +despl : 0
118 * LEXEMA : 'bisiesto'
119   ATRIBUTOS :
120     +tipo : 'funcion'
121     +numParam : 1
122     +TipoParam1 : 'constEnt'
123     +ModoParam1 : 1
124     +TipoRetorno : 'booleanR'
125     +EtiqFuncion : 'Etbisiesto01'
126 * LEXEMA : 'dias'
127   ATRIBUTOS :
128     +tipo : 'funcion'
129     +numParam : 2
130     +TipoParam1 : 'constEnt'
131     +ModoParam1 : 1
132     +TipoParam2 : 'constEnt'

```

```

133     +ModoParam2 : 1
134     +TipoRetorno : 'constEnt'
135     +EtiqFuncion : 'Etdias01'
136 * LEXEMA : 'esFechaCorrecta'
137     ATRIBUTOS :
138     +tipo : 'funcion'
139     +numParam : 3
140     +TipoParam1 : 'constEnt'
141     +ModoParam1 : 1
142     +TipoParam2 : 'constEnt'
143     +ModoParam2 : 1
144     +TipoParam3 : 'constEnt'
145     +ModoParam3 : 1
146     +TipoRetorno : 'booleanR'
147     +EtiqFuncion : 'EtesFechaCorrecta01'
148 * LEXEMA : 'demo'
149     ATRIBUTOS :
150     +tipo : 'funcion'
151     +numParam : 0
152     +TipoRetorno : 'vacio'
153     +EtiqFuncion : 'Etdemo01'
154 * LEXEMA : 'abc'
155     ATRIBUTOS :
156     +tipo : 'constEnt'
157     +despl : 1

```

En esta prueba se realiza todo tipo de operaciones, como sumas con llamadas a funciones, comparaciones con llamadas a funciones, switch sin break, anidamiento de funciones, entre otras cosas.