2012 | 2013

Analizador Léxico nADA

Procesadores de Lenguajes

Daniel Arbelo Cabrera
Alberto Manuel Mireles Suárez
David Guillermo Morales Sáez
Eduardo Quesada Díaz
María del Carmen Sánchez Medrano

Índice

Fichero lex

Para generar el analizador léxico se ha empleado la herramienta flex, la cual genera a partir de un fichero, el código en C del analizador léxico. A continuación se encuentra el contenido del fichero `lex` que define los tokens para el lenguaje nADA.

```
%x inc
%{
#define INT                 257
#define FLO                 258
#define CHAR                259
#define STRING              260

#define RES_PROCEDURE       261
#define RES_FUNCTION        262
#define RES_BEGIN           263
#define RES_END             264
#define RES_RETURN          265
#define RES_WITH            266
#define RES_IS              267
#define RES_IF              268
#define RES_ELSE            269
#define RES_THEN            270
#define RES_CASE            271
#define RES_WHEN            272
#define RES_LOOP            273
#define RES_WHILE           274
#define RES_INTEGER         275
#define RES_FLOAT           276
#define RES_CHAR            277
#define RES_STRING          278
#define RES_ARRAY           279
#define RES_IN              280
#define RES_OUT             281
#define RES_IN_OUT          282
#define RES_OF              283
#define RES_TYPE            284

#define OP_ASIGNACION       285

#define OP_MAS              286
#define OP_MENOS            287
#define OP_PRODUCTO         288
#define OP_DIVISION         289
#define OP_IGUAL            290
#define OP_MENOR            291
#define OP_MENORIGUAL       292
#define OP_MAYOR            293
#define OP_MAYORIGUAL       294
#define OP_NOT              295
#define OP_AND              296
#define OP_OR               297

#define SEP_PUNTOCOMA       298
#define SEP_COMA            299
#define SEP_ACORCHETE       300
#define SEP_CCORCHETE       301
```

```
#define SEP_APARENTESIS       302
#define SEP_CPARENTESIS       303
#define SEP_DOSPUNTOS         304
#define ID                    305
#define FICH                  306
#define RANGO                 307
#define OP_FLECHITA           308
#define OP_DISTINTO           309

int numlinea =1;
void error (char*);
%}

letra [a-zA-Z]
DD          [0-9]
PO          [1-9]
DEC   {PO}?{DD}+
simbolo     "<" | ">" | "<=" | ">=" | "=" | ":=" | "(" | ")" | "[" |
"]" |"+" | "-" | "_" | "*" | "/" | ";" | "," | "\" | "=>" | "/="
espacio " "
punto "."

%%
<<EOF>>     {
            yypop_buffer_state();
        if ( !YY_CURRENT_BUFFER )
        {
            yyterminate();
        }
        }

["\t" " "]  {/* Se ignoran espacios en blanco */}
"-""-".*    {/* Se ignoran comentarios de linea */}
\n                {numlinea++;}

procedure         {return RES_PROCEDURE;}
function          {return RES_FUNCTION;}
begin             {return RES_BEGIN;}
end               {return RES_END;}
return            {return RES_RETURN;}
with              {BEGIN(inc);}
is                {return RES_IS;}
if                {return RES_IF;}
else              {return RES_ELSE;}
then              {return RES_THEN;}
case              {return RES_CASE;}
when              {return RES_WHEN;}
loop              {return RES_LOOP;}
while             {return RES_WHILE;}
Integer           {return RES_INTEGER;}
float             {return RES_FLOAT;}
char              {return RES_CHAR;}
string            {return RES_STRING;}
array             {return RES_ARRAY;}
in                {return RES_IN;}
out               {return RES_OUT;}
inout             {return RES_IN_OUT;}
of                {return RES_OF;}
type              {return RES_TYPE;}
Fichero           {return FICH;}
```

```
"="">"            {return OP_FLECHITA;}
"/""="            {return OP_DISTINTO;}
".."              {return RANGO;}
"("               {return SEP_APARENTESIS;}
")"               {return SEP_CPARENTESIS;}
","               {return SEP_COMA;}
";"               {return SEP_PUNTOCOMA;}
"["               {return SEP_ACORCHETE;}
"]"               {return SEP_CCORCHETE;}
":"               {return SEP_DOSPUNTOS;}
"<"               {return OP_MENOR;}
">"               {return OP_MAYOR;}
"<""="            {return OP_MENORIGUAL;}
">""="            {return OP_MAYORIGUAL;}
and               {return OP_AND;}
or                {return OP_OR;}
not               {return OP_NOT;}
"="               {return OP_IGUAL;}
":""="            {return OP_ASIGNACION;}

{letra}({letra}|{PO}|_)*|(_)({letra}|{PO})({letra}|{PO}|_)*
{return ID;}

"+"               {return OP_MAS;}
"-"               {return OP_MENOS;}
"*"               {return OP_PRODUCTO;}
"/"               {return OP_DIVISION;}
{DEC}+            {return INT;}
{DEC}+"."{DEC}+   {return FLO;}

\"{letra}({DEC}|{letra}|{espacio}|{punto})*\"   {return STRING;}
\'({letra}|{DD}|{espacio})\'                    {return CHAR;}

<inc>[ \t]*             /* eat the whitespace */
<inc>[^ \t\n]+    {     /* got the include file name */
                  yyin = fopen( yytext, "r" );
                  if ( ! yyin ) error(yytext);
                  yypush_buffer_state(yy_create_buffer( yyin,
YY_BUF_SIZE ));
                  BEGIN(INITIAL);
                  }

.                 {error(yytext);}
%%


int main(int argc, char** argv)
{
      const char *simbolos[53];
      simbolos[0] = "INT";
      simbolos[1] = "FLO";
      simbolos[2] = "CHAR";
      simbolos[3] = "STRING";
      simbolos[4] = "RES_PROCEDURE";
      simbolos[5] = "RES_FUNCTION";
      simbolos[6] = "RES_BEGIN";
      simbolos[7] = "RES_END";
      simbolos[8] = "RES_RETURN";
      simbolos[9] = "RES_WITH";
```

```
	simbolos[10] = "RES_IS";
	simbolos[11] = "RES_IF";
	simbolos[12] = "RES_ELSE";
	simbolos[13] = "RES_THEN";
	simbolos[14] = "RES_CASE";
	simbolos[15] = "RES_WHEN";
	simbolos[16] = "RES_LOOP";
	simbolos[17] = "RES_WHILE";
	simbolos[18] = "RES_INTEGER";
	simbolos[19] = "RES_FLOAT";
	simbolos[20] = "RES_CHAR";
	simbolos[21] = "RES_STRING";
	simbolos[22] = "RES_ARRAY";
	simbolos[23] = "RES_IN";
	simbolos[24] = "RES_OUT";
	simbolos[25] = "RES_IN_OUT";
	simbolos[26] = "RES_OF";
	simbolos[27] = "RES_TYPE";
	simbolos[28] = "OP_ASIGNACION";
	simbolos[29] = "OP_MAS";
	simbolos[30] = "OP_MENOS";
	simbolos[31] = "OP_PRODUCTO";
	simbolos[32] = "OP_DIVISION";
	simbolos[33] = "OP_IGUAL";
	simbolos[34] = "OP_MENOR";
	simbolos[35] = "OP_MENORIGUAL";
	simbolos[36] = "OP_MAYOR";
	simbolos[37] = "OP_MAYORIGUAL";
	simbolos[38] = "OP_NOT";
	simbolos[39] = "OP_AND";
	simbolos[40] = "OP_OR";
	simbolos[41] = "SEP_PUNTOCOMA";
	simbolos[42] = "SEP_COMA";
	simbolos[43] = "SEP_ACORCHETE";
	simbolos[44] = "SEP_CCORCHETE";
	simbolos[45] = "SEP_APARENTESIS";
	simbolos[46] = "SEP_CPARENTESIS";
	simbolos[47] = "SEP_DOSPUNTOS";
	simbolos[48] = "ID";
	simbolos[49] = "FICH";
	simbolos[50] = "RANGO";
	simbolos[51] = "OP_FLECHITA";
	simbolos[52] = "OP_DISTINTO";


	int s;

	if (argc > 1)
	{
		printf("Abriendo fichero: %s\n", argv[1]);
		yyin = fopen(argv[1], "r");
		if (yyin == NULL)
			printf("\aError abriendo el fichero.\n");
		else
		{
			s = yylex();
			while (s != 0)
			{
				printf("%i: %s %s\n", s, simbolos[s-257],
yytext);
```

```
                        s = yylex();
                    }
                }
        }
        printf("\nANALISIS FINALIZADO\n");
        return 0;
}

void error(char* mens)
{
        printf("Error lexico en linea %i: %s\n", numlinea, mens);
}
```

Fichero Makefile

Para la compilación del fichero `lex.l`, se ha generado un archivo `makefile` el cual toma el fichero fuente, genera el código en C correspondiente mediante la herramienta `flex`, y posteriormente lo compila con `gcc` generando el ejecutable `alex`. Para la compilación con `gcc`, se han utilizado las opciones `-g`, que permite el modo depuración; `-Wall`, para mostrar todos los posibles avisos (warnings) relacionados con estructuras creadas por el usuario, entre otros; y `-lfl`, para indicar que el fichero a compilar se trata de un fuente generado por `flex`. También se incluye una opción `clean` para eliminar los ficheros generados. El fichero `makefile` en cuestión queda de la siguiente manera:

```
all: lex cc

lex:
        flex lex.l

cc:
        gcc -g -Wall lex.yy.c -o alex -lfl

clean:
        -rm lex.yy.c
        -rm alex
```

Pruebas

Se han creado ocho ficheros en los que se prueban cada una de las características del lenguaje, desde tipos de variables hasta estructuras de control. A continuación, se detallará cada fichero y el resultado devuelto por flex.

Fichero Prueba-Case

```
procedure EJEMPLOCASE (caracter: in Character; variable: out Integer) is
begin
    case caracter is
        when 'a' => variable := 0;
        when 'b' => variable := 1;
        when 'c' => variable := 2;
```

```
        when 'd' => variable := 3;
        when 'e' => variable := 4;
        when 'f' => variable := 5;
        when 'g' => variable := 6;
        when 'h' => variable := 7;
        when 'i' => variable := 8;
        when 'j' => variable := 9;
        when 'k' => variable := 10;
        when 'l' => variable := 11;
        when 'm' => variable := 12;
        when 'n' => variable := 13;
        when 'o' => variable := 14;
        when 'p' => variable := 15;
        when 'q' => variable := 16;
        when 'r' => variable := 17;
        when 's' => variable :=18 ;
        when 't' => variable := 19;
        when 'u' => variable := 20;
        when 'v' => variable := 21;
        when 'w' => variable := 22;
        when 'x' => variable := 23;
        when 'y' => variable := 24;
        when 'z' => variable := 25;
        when others => variable := -1;
    end case;
end EJEMPLOCASE;
```

Salida Prueba-Case

```
261: RES_PROCEDURE procedure
305: ID EJEMPLOCASE
302: SEP_APARENTESIS (
305: ID caracter
304: SEP_DOSPUNTOS :
280: RES_IN in
305: ID Character
298: SEP_PUNTOCOMA ;
305: ID variable
304: SEP_DOSPUNTOS :
281: RES_OUT out
275: RES_INTEGER Integer
303: SEP_CPARENTESIS )
267: RES_IS is
263: RES_BEGIN begin
271: RES_CASE case
305: ID caracter
267: RES_IS is
272: RES_WHEN when
259: CHAR 'a'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 0
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'b'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 1
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'c'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 2
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'd'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 3
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'e'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 4
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'f'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 5
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'g'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 6
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
```

```
259: CHAR 'h'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 7
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'i'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 8
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'j'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 9
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'k'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 10
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'l'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 11
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'm'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 12
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'n'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 13
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'o'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 14
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'p'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 15
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'q'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 16
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'r'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 17
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 's'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 18
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 't'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 19
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'u'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 20
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'v'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 21
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'w'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 22
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'x'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 23
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'y'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT 24
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
259: CHAR 'z'
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
```

```
257: INT 25
298: SEP_PUNTOCOMA ;
272: RES_WHEN when
305: ID others
308: OP_FLECHITA =>
305: ID variable
285: OP_ASIGNACION :=
257: INT -1
298: SEP_PUNTOCOMA ;
264: RES_END end
271: RES_CASE case
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLOCASE
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZADO
```

Fichero Prueba-Enumerado

```
procedure EJEMPLOENUM is
    type enumerado is (VALOR1, VALOR2, VALOR3, VALOR4);
    ejemplo : enumerado;
begin
    ejemplo := VALOR1;
    ejemplo := VALOR2;
    if(ejemplo = VALOR2) then
        Put("OK");
    end if;
end EJEMPLOENUM;
```

Salida Prueba-Enumerado

```
261: RES_PROCEDURE procedure
305: ID EJEMPLOENUM
267: RES_IS is
284: RES_TYPE type
305: ID enumerado
267: RES_IS is
302: SEP_APARENTESIS (
305: ID VALOR1
299: SEP_COMA ,
305: ID VALOR2
299: SEP_COMA ,
305: ID VALOR3
299: SEP_COMA ,
305: ID VALOR4
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
305: ID ejemplo
304: SEP_DOSPUNTOS :
305: ID enumerado
298: SEP_PUNTOCOMA ;
263: RES_BEGIN begin
305: ID ejemplo
285: OP_ASIGNACION :=
305: ID VALOR1
298: SEP_PUNTOCOMA ;
305: ID ejemplo
285: OP_ASIGNACION :=
305: ID VALOR2
298: SEP_PUNTOCOMA ;
268: RES_IF if
302: SEP_APARENTESIS (
305: ID ejemplo
290: OP_IGUAL =
305: ID VALOR2
303: SEP_CPARENTESIS )
270: RES_THEN then
305: ID Put
302: SEP_APARENTESIS (
260: STRING "OK"
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
264: RES_END end
268: RES_IF if
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLOENUM
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZADO
```

Fichero Text_Io.adb

Este fichero contiene las declaraciones prototipo para las operaciones sobre ficheros, y será incluido a través de la cláusula `with` en la prueba de ficheros. Las operaciones que contiene son `Open`, `Close`, `Get`, `Put`, y `End_Of_File`. El contenido de dicho fichero es el que sigue:

```
procedure Open (File : inout Integer;
        Mode : in String;
        Name : in String);

procedure Close (File : in Integer);

procedure Get (File : in Integer;
        Mode : out String);

procedure Put (File : in Integer;
        Mode : in String);

function End_Of_File (File : in Integer) return Integer;
```

Fichero Prueba-Ficheros

```
with Text_Io.adb

procedure EJEMPLOFICHERO (Fichero : in Text_Io.File_Type; caracter : out Character)
is
begin
    Open(Fichero, In_File, "fichero.txt");
    if(Is_Open(fichero)) then
        if(End_Of_File(Fichero)) then
            caracter := '0';
        else
            get(Fichero, caracter);
        end if;
    end if;
    Close(Fichero);

    Open(Fichero, Out_File, "fichero.txt");
    if(Is_Open(fichero)) then
        put(Fichero, caracter);
    end if;
    Close(Fichero);

    Open(Fichero, Append_File, "fichero.txt");
    if(Is_Open(fichero)) then
        put(Fichero, caracter);
    end if;
    Close(Fichero);
end EJEMPLOFICHERO;
```

Salida Prueba-Ficheros

```
261: RES_PROCEDURE procedure
305: ID Open
302: SEP_APARENTESIS (
305: ID File
304: SEP_DOSPUNTOS :
280: RES_IN in
281: RES_OUT out
275: RES_INTEGER Integer
298: SEP_PUNTOCOMA ;
305: ID Mode
304: SEP_DOSPUNTOS :
280: RES_IN in
305: ID String
298: SEP_PUNTOCOMA ;
305: ID Name
304: SEP_DOSPUNTOS :
280: RES_IN in
305: ID String
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
261: RES_PROCEDURE procedure
305: ID Close
302: SEP_APARENTESIS (
305: ID File
304: SEP_DOSPUNTOS :
280: RES_IN in
275: RES_INTEGER Integer
303: SEP_CPARENTESIS )
```

```
298: SEP_PUNTOCOMA ;
261: RES_PROCEDURE procedure
305: ID Get
302: SEP_APARENTESIS (
305: ID File
304: SEP_DOSPUNTOS :
280: RES_IN in
275: RES_INTEGER Integer
298: SEP_PUNTOCOMA ;
305: ID Mode
304: SEP_DOSPUNTOS :
281: RES_OUT out
305: ID String
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
261: RES_PROCEDURE procedure
305: ID Put
302: SEP_APARENTESIS (
305: ID File
304: SEP_DOSPUNTOS :
280: RES_IN in
275: RES_INTEGER Integer
298: SEP_PUNTOCOMA ;
305: ID Mode
304: SEP_DOSPUNTOS :
280: RES_IN in
305: ID String
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
262: RES_FUNCTION function
305: ID End_Of_File
302: SEP_APARENTESIS (
305: ID File
304: SEP_DOSPUNTOS :
280: RES_IN in
275: RES_INTEGER Integer
303: SEP_CPARENTESIS )
265: RES_RETURN return
275: RES_INTEGER Integer
298: SEP_PUNTOCOMA ;
261: RES_PROCEDURE procedure
305: ID EJEMPLOFICHERO
302: SEP_APARENTESIS (
306: FICH Fichero
304: SEP_DOSPUNTOS :
280: RES_IN in
305: ID Text_Io
Error lexico en linea 24: .
305: ID File_Type
298: SEP_PUNTOCOMA ;
305: ID caracter
304: SEP_DOSPUNTOS :
281: RES_OUT out
305: ID Character
303: SEP_CPARENTESIS )
267: RES_IS is
263: RES_BEGIN begin
305: ID Open
302: SEP_APARENTESIS (
306: FICH Fichero
299: SEP_COMA ,
305: ID In_File
299: SEP_COMA ,
260: STRING "fichero.txt"
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
268: RES_IF if
302: SEP_APARENTESIS (
305: ID Is_Open
302: SEP_APARENTESIS (
305: ID fichero
303: SEP_CPARENTESIS )
303: SEP_CPARENTESIS )
270: RES_THEN then
268: RES_IF if
302: SEP_APARENTESIS (
305: ID End_Of_File
302: SEP_APARENTESIS (
306: FICH Fichero
303: SEP_CPARENTESIS )
303: SEP_CPARENTESIS )
270: RES_THEN then
305: ID caracter
285: OP_ASIGNACION :=
259: CHAR '0'
298: SEP_PUNTOCOMA ;
269: RES_ELSE else
305: ID get
302: SEP_APARENTESIS (
306: FICH Fichero
299: SEP_COMA ,
305: ID caracter
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
264: RES_END end
268: RES_IF if
298: SEP_PUNTOCOMA ;
264: RES_END end
268: RES_IF if
298: SEP_PUNTOCOMA ;
305: ID Close
302: SEP_APARENTESIS (
306: FICH Fichero
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
305: ID Open
302: SEP_APARENTESIS (
306: FICH Fichero
299: SEP_COMA ,
305: ID Out_File
299: SEP_COMA ,
260: STRING "fichero.txt"
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
268: RES_IF if
302: SEP_APARENTESIS (
305: ID Is_Open
302: SEP_APARENTESIS (
305: ID fichero
303: SEP_CPARENTESIS )
303: SEP_CPARENTESIS )
270: RES_THEN then
305: ID put
302: SEP_APARENTESIS (
306: FICH Fichero
299: SEP_COMA ,
305: ID caracter
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
264: RES_END end
```

```
268: RES_IF if
298: SEP_PUNTOCOMA ;
305: ID Close
302: SEP_APARENTESIS (
306: FICH Fichero
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
305: ID Open
302: SEP_APARENTESIS (
306: FICH Fichero
299: SEP_COMA ,
305: ID Append_File
299: SEP_COMA ,
260: STRING "fichero.txt"
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
268: RES_IF if
302: SEP_APARENTESIS (
305: ID Is_Open
302: SEP_APARENTESIS (
305: ID fichero
303: SEP_CPARENTESIS )
303: SEP_CPARENTESIS )
270: RES_THEN then
305: ID put
302: SEP_APARENTESIS (
306: FICH Fichero
299: SEP_COMA ,
305: ID caracter
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
264: RES_END end
268: RES_IF if
298: SEP_PUNTOCOMA ;
305: ID Close
302: SEP_APARENTESIS (
306: FICH Fichero
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLOFICHERO
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZADO
```

Fichero Prueba-Funciones

```
procedure EJEMPLOFUNC (salida: inout Integer) is

    function sumador (p1: Integer; p2: Integer) return Integer is
    begin
        return p1+p2;
    end sumador;
begin
    salida := 2*sumador(2, 3);
end EJEMPLOFUNC;
```

Salida Prueba-Funciones

```
261: RES_PROCEDURE procedure
305: ID EJEMPLOFUNC
302: SEP_APARENTESIS (
305: ID salida
304: SEP_DOSPUNTOS :
282: RES_IN_OUT inout
275: RES_INTEGER Integer
303: SEP_CPARENTESIS )
267: RES_IS is
262: RES_FUNCTION function
305: ID sumador
302: SEP_APARENTESIS (
305: ID p1
304: SEP_DOSPUNTOS :
275: RES_INTEGER Integer
298: SEP_PUNTOCOMA ;
305: ID p2
304: SEP_DOSPUNTOS :
275: RES_INTEGER Integer
303: SEP_CPARENTESIS )
265: RES_RETURN return
275: RES_INTEGER Integer
267: RES_IS is
263: RES_BEGIN begin
265: RES_RETURN return
305: ID p1
286: OP_MAS +
305: ID p2
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID sumador
298: SEP_PUNTOCOMA ;
263: RES_BEGIN begin
305: ID salida
285: OP_ASIGNACION :=
257: INT 2
288: OP_PRODUCTO *
305: ID sumador
302: SEP_APARENTESIS (
257: INT 2
299: SEP_COMA ,
257: INT 3
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLOFUNC
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZADO
```

Fichero Prueba-If

```
procedure EJEMPLOIF (variable: inout Integer) is
begin
    if(variable = 8) then
        variable := 1;
    else
        variable := 0;
    end if;
end EJEMPLOIF;
```

Salida Prueba-If

```
261: RES_PROCEDURE procedure
305: ID EJEMPLOIF
302: SEP_APARENTESIS (
305: ID variable
304: SEP_DOSPUNTOS :
282: RES_IN_OUT inout
275: RES_INTEGER Integer
303: SEP_CPARENTESIS )
267: RES_IS is
263: RES_BEGIN begin
268: RES_IF if
302: SEP_APARENTESIS (
305: ID variable
290: OP_IGUAL =
257: INT 8
303: SEP_CPARENTESIS )
270: RES_THEN then
305: ID variable
285: OP_ASIGNACION :=
257: INT 1
298: SEP_PUNTOCOMA ;
269: RES_ELSE else
305: ID variable
285: OP_ASIGNACION :=
257: INT 0
298: SEP_PUNTOCOMA ;
264: RES_END end
268: RES_IF if
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLOIF
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZADO
```

Fichero Prueba-Ristra

```
procedure EJEMPLORISTRA is
    ristra : String;
begin
    ristra := "Esto es una ristra";
    Put(ristra);
end EJEMPLORISTRA;
```

Salida Prueba-Ristra

```
261: RES_PROCEDURE procedure
305: ID EJEMPLORISTRA
267: RES_IS is
305: ID ristra
304: SEP_DOSPUNTOS :
305: ID String
298: SEP_PUNTOCOMA ;
263: RES_BEGIN begin
305: ID ristra
285: OP_ASIGNACION :=
260: STRING "Esto es una ristra"
298: SEP_PUNTOCOMA ;
305: ID Put
302: SEP_APARENTESIS (
305: ID ristra
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLORISTRA
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZADO
```

Fichero Prueba-Vector

```
procedure EJEMPLOVECTOR is
    type Array_Entero is array (1..10) of Integer;
    vector : Array_Entero;
    contador : Integer;
begin
    contador := 0;
    while(contador < 10) loop
        vector(contador) := contador;
        contador := contador +1;
    end loop;
    contador := 0;
    while(contador < 10) loop
        Put(vector(contador));
        contador := contador +1;
    end loop;
end EJEMPLOVECTOR;
```

Salida Prueba-Vector

```
261: RES_PROCEDURE procedure
305: ID EJEMPLOVECTOR
267: RES_IS is
284: RES_TYPE type
305: ID Array_Entero
267: RES_IS is
279: RES_ARRAY array
302: SEP_APARENTESIS (
257: INT 1
307: RANGO ..
257: INT 10
303: SEP_CPARENTESIS )
283: RES_OF of
275: RES_INTEGER Integer
298: SEP_PUNTOCOMA ;
305: ID vector
304: SEP_DOSPUNTOS :
305: ID Array_Entero
298: SEP_PUNTOCOMA ;
305: ID contador
304: SEP_DOSPUNTOS :
275: RES_INTEGER Integer
298: SEP_PUNTOCOMA ;
263: RES_BEGIN begin
305: ID contador
285: OP_ASIGNACION :=
257: INT 0
298: SEP_PUNTOCOMA ;
274: RES_WHILE while
302: SEP_APARENTESIS (
305: ID contador
291: OP_MENOR <
257: INT 10
303: SEP_CPARENTESIS )
273: RES_LOOP loop
305: ID vector
302: SEP_APARENTESIS (
305: ID contador
303: SEP_CPARENTESIS )
285: OP_ASIGNACION :=
305: ID contador
298: SEP_PUNTOCOMA ;
305: ID contador
285: OP_ASIGNACION :=
305: ID contador
286: OP_MAS +
257: INT 1
298: SEP_PUNTOCOMA ;
264: RES_END end
273: RES_LOOP loop
298: SEP_PUNTOCOMA ;
305: ID contador
285: OP_ASIGNACION :=
257: INT 0
298: SEP_PUNTOCOMA ;
274: RES_WHILE while
302: SEP_APARENTESIS (
305: ID contador
291: OP_MENOR <
257: INT 10
303: SEP_CPARENTESIS )
273: RES_LOOP loop
305: ID Put
302: SEP_APARENTESIS (
305: ID vector
302: SEP_APARENTESIS (
305: ID contador
303: SEP_CPARENTESIS )
303: SEP_CPARENTESIS )
298: SEP_PUNTOCOMA ;
305: ID contador
285: OP_ASIGNACION :=
305: ID contador
286: OP_MAS +
257: INT 1
298: SEP_PUNTOCOMA ;
264: RES_END end
273: RES_LOOP loop
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLOVECTOR
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZADO
```

Fichero Prueba-While

```
procedure EJEMPLOWHILE (variable : inout Integer) is
begin
    while ( variable /= 0) loop
        variable := variable -1;
    end loop;
end EJEMPLOWHILE;
```

Salida Prueba-While

```
261: RES_PROCEDURE procedure
305: ID EJEMPLOWHILE
302: SEP_APARENTESIS (
305: ID variable
304: SEP_DOSPUNTOS :
282: RES_IN_OUT inout
275: RES_INTEGER Integer
303: SEP_CPARENTESIS )
267: RES_IS is
263: RES_BEGIN begin
274: RES_WHILE while
302: SEP_APARENTESIS (
305: ID variable
309: OP_DISTINTO /=
257: INT 0
303: SEP_CPARENTESIS )
273: RES_LOOP loop
305: ID variable
285: OP_ASIGNACION :=
305: ID variable
287: OP_MENOS -
257: INT 1
298: SEP_PUNTOCOMA ;
264: RES_END end
273: RES_LOOP loop
298: SEP_PUNTOCOMA ;
264: RES_END end
305: ID EJEMPLOWHILE
298: SEP_PUNTOCOMA ;

ANALISIS FINALIZAD
```