2012 | 2013

# Analizador Sintáctico nADA

## Procesadores de Lenguajes

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

Universidad de Las Palmas de Gran Canaria

## Índice

## Fichero lex.l

En esta fase se ha modificado ligeramente este fichero con el fin de facilitar el análisis sintáctico reduciendo conflictos en la gramática. La modificación ha sido la de exigir que los tipos de variables definidas en el lenguaje empiecen por una letra mayúscula (Integer, Float, Char, String, Array). A su vez, se ha modificado un aspecto del lenguaje, de manera que ahora el índice de un vector se indica entre corchetes, en vez de con paréntesis.

```
%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);}
%%


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

## Fichero sint.y

```
%{

#include <stdio.h>
extern FILE *yyin;
extern int numlin;
int yydebug=1;

%}

%token INT
%token FLO
%token CHAR
%token STRING

%token RES_PROCEDURE
%token RES_FUNCTION
%token RES_BEGIN
%token RES_END
%token RES_RETURN
%token RES_IS
%token RES_IF
%token RES_ELSE
%token RES_THEN
%token RES_CASE
%token RES_WHEN
%token RES_LOOP
%token RES_WHILE
%token RES_INTEGER
%token RES_FLOAT
%token RES_CHAR
%token RES_STRING
%token RES_ARRAY
%token RES_IN
%token RES_OUT
%token RES_IN_OUT
%token RES_OF
%token RES_TYPE

%token OP_ASIGNACION

%token OP_MAS
%token OP_MENOS
%token OP_PRODUCTO
%token OP_DIVISION
%token OP_IGUAL
%token OP_MENOR
%token OP_MENORIGUAL
%token OP_MAYOR
%token OP_MAYORIGUAL
%token OP_NOT
%token OP_AND
%token OP_OR

%token SEP_PUNTOCOMA
%token SEP_COMA
%token SEP_APARENTESIS
%token SEP_CPARENTESIS
%token SEP_ACORCHETE
%token SEP_CCORCHETE
%token SEP_DOSPUNTOS
%token ID
```

```
%token FICH
%token RANGO
%token OP_FLECHITA
%token OP_DISTINTO

%left OP_AND
%left OP_OR
%left OP_NOT
%left OP_MENOR
%left OP_MENORIGUAL
%left OP_MAYOR
%left OP_MAYORIGUAL
%left OP_PRODUCTO
%left OP_DIVISION
%left OP_MAS
%left OP_MENOS
%start program

%%

program : RES_PROCEDURE ID SEP_APARENTESIS procedure_param SEP_CPARENTESIS RES_IS
declaraciones
          RES_BEGIN sequence_of_statements RES_END ID SEP_PUNTOCOMA
          {printf("%d :: <program> RES_PROCEDURE \n",numlin);}
          | RES_PROCEDURE ID SEP_APARENTESIS procedure_param SEP_CPARENTESIS
SEP_PUNTOCOMA program  {printf("%d :: <program> Prototipo_proc \n",numlin);}
          | RES_FUNCTION ID SEP_APARENTESIS function_param SEP_CPARENTESIS RES_RETURN
types SEP_PUNTOCOMA program  {printf("%d :: <program> Prototipo_func \n",numlin);}
          ;

procedure_call_statement : ID SEP_APARENTESIS parametros SEP_CPARENTESIS
SEP_PUNTOCOMA
                           {printf("%d :: <procedure_call_statement> ID...
\n",numlin);}
                         ;

function_call_statement : ID SEP_APARENTESIS parametros SEP_CPARENTESIS
                          {printf("%d :: <function_call_statement> ID..\n",numlin);}
                        ;

subprogram_declaration : subprogram_specification SEP_PUNTOCOMA
                         {printf("%d :: <subprogram_declaration>
subprogram_specification \n",numlin);}
                       ;

subprogram_specification : RES_PROCEDURE ID SEP_APARENTESIS procedure_param
SEP_CPARENTESIS
                           RES_IS declaraciones RES_BEGIN sequence_of_statements
RES_END ID
                           {printf("%d :: <subprogram_specification> RES_PROCEDURE
\n",numlin);}
                         | RES_FUNCTION ID SEP_APARENTESIS function_param
SEP_CPARENTESIS RES_RETURN types
                           RES_IS declaraciones RES_BEGIN sequenciafunction RES_END
ID
                           {printf("%d :: <subprogram_specification> RES_FUNCTION
\n",numlin);}
                         ;

procedure_param : procedure_param SEP_PUNTOCOMA ID SEP_DOSPUNTOS tipo_ent_proc types
                  {printf("%d :: <procedure_param> procedure_param SEP_PUNTOCOMA
\n",numlin);}
                | ID SEP_DOSPUNTOS tipo_ent_proc types
                  {printf("%d :: <procedure_param> ID SEP_DOSPUNTOS \n",numlin);}
                | {printf("%d :: <procedure_param> empty \n",numlin);}
                ;

function_param : function_param SEP_PUNTOCOMA ID SEP_DOSPUNTOS types
                 {printf("%d :: <function_param> function_param SEP_PUNTOCOMA
\n",numlin);}
               | ID SEP_DOSPUNTOS types
                 {printf("%d :: <function_param> ID SEP_DOSPUNTOS \n",numlin);}
               | {printf("%d :: <function_param> empty \n",numlin);}
               ;

tipo_ent_proc : RES_IN       {printf("%d :: <tipo_ent_proc> RES_IN \n",numlin);}
              | RES_OUT      {printf("%d :: <tipo_ent_proc> RES_OUT \n",numlin);}
              | RES_IN_OUT   {printf("%d :: <tipo_ent_proc> RES_IN_OUT \n",numlin);}
              ;

declaraciones : declaraciones declaracion {printf("%d :: <declaraciones>
declaraciones declaracion \n",numlin);}
              |   {printf("%d :: <declaraciones> empty \n",numlin);}
              ;

declaracion : ID SEP_DOSPUNTOS types SEP_PUNTOCOMA  {printf("%d :: <declaracion> ID
SEP_DOSPUNTOS \n",numlin);}
            | enum_type_def  {printf("%d :: <declaracion> enum_type_def \n",numlin);}
            | array_def      {printf("%d :: <declaracion> array_def \n",numlin);}
            | subprogram_declaration  {printf("%d :: <declaracion>
subprogram_declaration\n",numlin);}
            ;

parametros : var                      {printf("%d :: <parametros> var \n",numlin);}
           | parametros SEP_COMA var  {printf("%d :: <parametros> parametros SEP_COMA
\n",numlin);}
           |                          {printf("%d :: <parametros> empty \n",numlin);}
           ;

var : ID                              {printf("%d :: <var> ID \n",numlin);}
    | INT                             {printf("%d :: <var> INT \n",numlin);}
    | FLO                             {printf("%d :: <var> FLO \n",numlin);}
    | CHAR                            {printf("%d :: <var> CHAR \n",numlin);}
    | STRING                          {printf("%d :: <var> STRING \n",numlin);}
    | ID SEP_ACORCHETE expression SEP_CCORCHETE {printf("%d :: <var> vector
\n",numlin);}
    ;

sequenciafunction : sequence_of_statements   {printf("%d :: <sequenciafunction>
sequence_of_statements \n",numlin);}
                  ;

sequence_of_statements : sequence_of_statements statement
                         {printf("%d :: <sequence_of_statements>
sequence_of_statements statement \n",numlin);}
                       | statement
                         {printf("%d :: <sequence_of_statements> statement
\n",numlin);}
                       ;

statement : simple_statement    {printf("%d :: <statement> simple_statement
\n",numlin);}
          | compound_statement  {printf("%d :: <statement> compound_statement
\n",numlin);}
          ;

compound_statement : if_statement  {printf("%d :: <compound_statement> if_statement
\n",numlin);}
                   | case_statement    {printf("%d :: <compound_statement>
case_statement \n",numlin);}
```

```
                   | while_statement   {printf("%d :: <compound_statement>
while_statement \n",numlin);}
                   | asignment_statement             {printf("%d ::
<compound_statement> asignment_statement \n",numlin);}
                   | return_statement                {printf("%d ::
<compound_statement> return_statement \n",numlin);}
                   ;

simple_statement : procedure_call_statement          {printf("%d ::
<simple_statement> procedure_call_statement \n",numlin);}
                 ;

asignment_statement : ID OP_ASIGNACION expression SEP_PUNTOCOMA
                      {printf("%d :: <asignment_statement> ID OP_ASIGNACION
\n",numlin);}
                    | ID OP_ASIGNACION STRING SEP_PUNTOCOMA
                      {printf("%d :: <asignment_statement> STRING \n",numlin);}
                    | ID SEP_ACORCHETE expression SEP_CCORCHETE OP_ASIGNACION
expression SEP_PUNTOCOMA
                      {printf("%d :: <asignment_statement> vector \n",numlin);}
                    | ID OP_ASIGNACION CHAR SEP_PUNTOCOMA
                      {printf("%d :: <asignment_statement> char \n",numlin);}
                    ;

case_statement : RES_CASE factor RES_IS case_statement_alternative alternative2
RES_END RES_CASE SEP_PUNTOCOMA
                 {printf("%d :: <case_statement> RES_CASE ID \n",numlin);}
               ;
case_statement_alternative : RES_WHEN var OP_FLECHITA sequence_of_statements
                             {printf("%d :: <case_statement_alternative> RES_WHEN var
\n",numlin);}
                           ;

alternative2 : alternative2 case_statement_alternative
               {printf("%d :: <alternative2> alternative2 case_statement_alternative
\n",numlin);}
             | {printf("%d :: <alternative2> empty \n",numlin);}
             ;

while_statement : RES_WHILE SEP_APARENTESIS cond_expression SEP_CPARENTESIS RES_LOOP
                  sequence_of_statements RES_END RES_LOOP  SEP_PUNTOCOMA
                  {printf("%d :: <while_statement> RES_WHILE SEP_APARENTESIS
\n",numlin);}
                ;

if_statement : RES_IF SEP_APARENTESIS cond_expression SEP_CPARENTESIS RES_THEN
               sequence_of_statements if_stat2 RES_END RES_IF SEP_PUNTOCOMA
               {printf("%d :: <if_statement> RES_IF SEP_APARENTESIS \n",numlin);}
             ;

if_stat2 : RES_ELSE sequence_of_statements           {printf("%d :: <if_stat2>
RES_ELSE sequence_of_statements \n",numlin);}
         |                                           {printf("%d :: <if_stat2> empty
\n",numlin);}
         ;

return_statement : RES_RETURN math_expression SEP_PUNTOCOMA
                   {printf("%d :: <return_statement> RES_RETURN math_expression
\n",numlin);}
                 | RES_RETURN CHAR SEP_PUNTOCOMA
                   {printf("%d :: <return_statement> RES_RETURN CHAR \n",numlin);}
                 | RES_RETURN STRING SEP_PUNTOCOMA
                   {printf("%d :: <return_statement> RES_RETURN STRING \n",numlin);}
                 ;
```

```
enum_type_def : RES_TYPE ID RES_IS SEP_APARENTESIS enum_type_def2 SEP_CPARENTESIS
SEP_PUNTOCOMA
                {printf("%d :: <enum_type_def> RES_TYPE ID \n",numlin);}
              ;

enum_type_def2 : enum_type_def2 SEP_COMA enum_lit_spec
                 {printf("%d :: <enum_type_def2> enum_type_def2 \n",numlin);}
               | enum_lit_spec
                 {printf("%d :: <enum_type_def2> enum_lit_spec \n",numlin);}
               ;

enum_lit_spec : ID          {printf("%d :: <enum_lit_spec> ID \n",numlin);}
              | CHAR        {printf("%d :: <enum_lit_spec> CHAR \n",numlin);}
              ;

array_def : RES_TYPE ID RES_IS RES_ARRAY SEP_APARENTESIS INT RANGO INT
SEP_CPARENTESIS RES_OF types SEP_PUNTOCOMA
            {printf("%d :: <array_def> RES_TYPE ID \n",numlin);}
          ;

types : ID                          {printf("%d :: <types> ID \n",numlin);}
      | RES_INTEGER                 {printf("%d :: <types> RES_INTEGER \n",numlin);}
      | RES_FLOAT                   {printf("%d :: <types> RES_FLOAT \n",numlin);}
      | RES_CHAR                    {printf("%d :: <types> RES_CHAR \n",numlin);}
      | RES_STRING                  {printf("%d :: <types> RES_STRING \n",numlin);}
      | FICH                        {printf("%d :: <types> FICH \n",numlin);}
      ;


additive_operation : OP_MAS  {printf("%d :: <additive_operation> OP_MAS \n",numlin);}
                   | OP_MENOS  {printf("%d :: <additive_operation> OP_MENOS
\n",numlin);}
                   ;

multiplicative_operation : OP_PRODUCTO               {printf("%d ::
<multiplicative_operation> OP_PRODUCTO \n",numlin);}
                         | OP_DIVISION               {printf("%d ::
<multiplicative_operation> OP_DIVISION \n",numlin);}
                         ;

cond_op : OP_IGUAL               {printf("%d :: <cond_op> OP_IGUAL \n",numlin);}
        | OP_MENOR               {printf("%d :: <cond_op> OP_MENOR \n",numlin);}
        | OP_MENORIGUAL          {printf("%d :: <cond_op> OP_MENORIGUAL \n",numlin);}
        | OP_MAYOR               {printf("%d :: <cond_op> OP_MAYOR \n",numlin);}
        | OP_MAYORIGUAL          {printf("%d :: <cond_op> OP_MAYORIGUAL \n",numlin);}
        | OP_DISTINTO            {printf("%d :: <cond_op> OP_DISTINTO \n",numlin);}
        ;

expression : math_expression {printf("%d :: <expression> math_expression
\n",numlin);}
           ;

cond_expression : cond_expression OP_OR conditional_expression
                  {printf("%d :: <cond_expression> cond_expression OP_OR
\n",numlin);}
                | cond_expression OP_AND conditional_expression
                  {printf("%d :: <cond_expression> cond_expression OP_AND
\n",numlin);}
                | conditional_expression
                  {printf("%d :: <cond_expression> cond_expression \n",numlin);}
                | OP_NOT conditional_expression
                  {printf("%d :: <cond_expression> cond_expression OP_NOT
\n",numlin);}
                ;
```

```
conditional_expression : conditional_expression cond_op math_expression
                         {printf("%d :: <conditional_expression>
conditional_expression \n",numlin);}
                       | math_expression
                         {printf("%d :: <conditional_expression> math_expression
\n",numlin);}
                       ;

operation_expression : operation_expression multiplicative_operation factor
                       {printf("%d :: <operation_expression> operation_expression
\n",numlin);}
                     | factor
                       {printf("%d :: <operation_expression> factor \n",numlin);}
                     | function_call_statement
                       {printf("%d :: <operation_expression> function_call_statement
\n",numlin);}
                     | operation_expression multiplicative_operation
function_call_statement
                       {printf("%d :: <operation_expression> operation_expression
multiplicative_operation function_call_statement\n",numlin);}
                     ;

math_expression : math_expression additive_operation operation_expression
                  {printf("%d :: <math_expression> math_expression \n",numlin);}
                | operation_expression
                  {printf("%d :: <math_expression> operation_expression \n",numlin);}
                | OP_MENOS operation_expression
                  {printf("%d :: <math_expression> OP_MENOS \n",numlin);}
                ;

factor : SEP_APARENTESIS expression SEP_CPARENTESIS  {printf("%d :: <factor>
SEP_APARENTESIS \n",numlin);}
       | ID                                    {printf("%d :: <factor> ID \n",numlin);}
       | INT                                   {printf("%d :: <factor> INT \n",numlin);}
       | FLO                                   {printf("%d :: <factor> FLO \n",numlin);}
       | ID SEP_ACORCHETE expression SEP_CCORCHETE   {printf("%d :: <factor> vector
\n",numlin);}
       ;

%%

int main()
{
    yyparse();
}

void yyerror (char *s)
{
    printf ("Error en linea %d: %s\n",numlin, s);
}
```

## Script de compilación

Para generar el analizador sintáctico, se ha empleado el siguiente script:

```
bison -v -d -t sint.y
flex lex.l
gcc -o sint sint.tab.c lex.yy.c -lfl
```

## 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 el analizador sintáctico. Los ficheros de salida generados por el analizador sintáctico en los que figuran todos los estados por el que pasa se adjuntan a la memoria.

### Fichero Prueba-Case

```
procedure EJEMPLOCASE (caracter: in Char; 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;
    end case;
end EJEMPLOCASE;
```

### Salida Prueba-Case

```
1 :: <tipo_ent_proc> RES_IN
1 :: <types> RES_CHAR
1 :: <procedure_param> ID SEP_DOSPUNTOS
1 :: <tipo_ent_proc> RES_OUT
1 :: <types> RES_INTEGER
1 :: <procedure_param> procedure_param SEP_PUNTOCOMA
1 :: <declaraciones> empty
3 :: <factor> ID
4 :: <var> CHAR
4 :: <factor> INT
4 :: <operation_expression> factor
4 :: <math_expression> operation_expression
4 :: <expression> math_expression
4 :: <asignment_statement> ID OP_ASIGNACION
4 :: <compound_statement> asignment_statement
4 :: <statement> compound_statement
```

```
4 :: <sequence_of_statements> statement
5 :: <case_statement_alternative> RES_WHEN var
5 :: <alternative2> empty
5 :: <var> CHAR
5 :: <factor> INT
5 :: <operation_expression> factor
5 :: <math_expression> operation_expression
5 :: <expression> math_expression
5 :: <asignment_statement> ID OP_ASIGNACION
5 :: <compound_statement> asignment_statement
5 :: <statement> compound_statement
5 :: <sequence_of_statements> statement
6 :: <case_statement_alternative> RES_WHEN var
6 :: <alternative2> alternative2 case_statement_alternative
6 :: <var> CHAR
6 :: <factor> INT
6 :: <operation_expression> factor
6 :: <math_expression> operation_expression
6 :: <expression> math_expression
6 :: <asignment_statement> ID OP_ASIGNACION
6 :: <compound_statement> asignment_statement
6 :: <statement> compound_statement
6 :: <sequence_of_statements> statement
7 :: <case_statement_alternative> RES_WHEN var
7 :: <alternative2> alternative2 case_statement_alternative
7 :: <var> CHAR
7 :: <factor> INT
7 :: <operation_expression> factor
7 :: <math_expression> operation_expression
7 :: <expression> math_expression
7 :: <asignment_statement> ID OP_ASIGNACION
7 :: <compound_statement> asignment_statement
7 :: <statement> compound_statement
7 :: <sequence_of_statements> statement
8 :: <case_statement_alternative> RES_WHEN var
8 :: <alternative2> alternative2 case_statement_alternative
8 :: <var> CHAR
8 :: <factor> INT
8 :: <operation_expression> factor
8 :: <math_expression> operation_expression
8 :: <expression> math_expression
8 :: <asignment_statement> ID OP_ASIGNACION
8 :: <compound_statement> asignment_statement
8 :: <statement> compound_statement
8 :: <sequence_of_statements> statement
9 :: <case_statement_alternative> RES_WHEN var
9 :: <alternative2> alternative2 case_statement_alternative
9 :: <var> CHAR
9 :: <factor> INT
9 :: <operation_expression> factor
9 :: <math_expression> operation_expression
9 :: <expression> math_expression
9 :: <asignment_statement> ID OP_ASIGNACION
9 :: <compound_statement> asignment_statement
9 :: <statement> compound_statement
9 :: <sequence_of_statements> statement
10 :: <case_statement_alternative> RES_WHEN var
10 :: <alternative2> alternative2 case_statement_alternative
10 :: <var> CHAR
10 :: <factor> INT
10 :: <operation_expression> factor
10 :: <math_expression> operation_expression
10 :: <expression> math_expression
10 :: <asignment_statement> ID OP_ASIGNACION
10 :: <compound_statement> asignment_statement
```

```
10 :: <statement> compound_statement
10 :: <sequence_of_statements> statement
11 :: <case_statement_alternative> RES_WHEN var
11 :: <alternative2> alternative2 case_statement_alternative
11 :: <var> CHAR
11 :: <factor> INT
11 :: <operation_expression> factor
11 :: <math_expression> operation_expression
11 :: <expression> math_expression
11 :: <asignment_statement> ID OP_ASIGNACION
11 :: <compound_statement> asignment_statement
11 :: <statement> compound_statement
11 :: <sequence_of_statements> statement
12 :: <case_statement_alternative> RES_WHEN var
12 :: <alternative2> alternative2 case_statement_alternative
12 :: <var> CHAR
12 :: <factor> INT
12 :: <operation_expression> factor
12 :: <math_expression> operation_expression
12 :: <expression> math_expression
12 :: <asignment_statement> ID OP_ASIGNACION
12 :: <compound_statement> asignment_statement
12 :: <statement> compound_statement
12 :: <sequence_of_statements> statement
13 :: <case_statement_alternative> RES_WHEN var
13 :: <alternative2> alternative2 case_statement_alternative
13 :: <var> CHAR
13 :: <factor> INT
13 :: <operation_expression> factor
13 :: <math_expression> operation_expression
13 :: <expression> math_expression
13 :: <asignment_statement> ID OP_ASIGNACION
13 :: <compound_statement> asignment_statement
13 :: <statement> compound_statement
13 :: <sequence_of_statements> statement
14 :: <case_statement_alternative> RES_WHEN var
14 :: <alternative2> alternative2 case_statement_alternative
14 :: <var> CHAR
14 :: <factor> INT
14 :: <operation_expression> factor
14 :: <math_expression> operation_expression
14 :: <expression> math_expression
14 :: <asignment_statement> ID OP_ASIGNACION
14 :: <compound_statement> asignment_statement
14 :: <statement> compound_statement
14 :: <sequence_of_statements> statement
15 :: <case_statement_alternative> RES_WHEN var
15 :: <alternative2> alternative2 case_statement_alternative
15 :: <var> CHAR
15 :: <factor> INT
15 :: <operation_expression> factor
15 :: <math_expression> operation_expression
15 :: <expression> math_expression
15 :: <asignment_statement> ID OP_ASIGNACION
15 :: <compound_statement> asignment_statement
15 :: <statement> compound_statement
15 :: <sequence_of_statements> statement
16 :: <case_statement_alternative> RES_WHEN var
16 :: <alternative2> alternative2 case_statement_alternative
16 :: <var> CHAR
16 :: <factor> INT
16 :: <operation_expression> factor
16 :: <math_expression> operation_expression
16 :: <expression> math_expression
16 :: <asignment_statement> ID OP_ASIGNACION
```

```
16 :: <compound_statement> asignment_statement
16 :: <statement> compound_statement
16 :: <sequence_of_statements> statement
17 :: <case_statement_alternative> RES_WHEN var
17 :: <alternative2> alternative2 case_statement_alternative
17 :: <var> CHAR
17 :: <factor> INT
17 :: <operation_expression> factor
17 :: <math_expression> operation_expression
17 :: <expression> math_expression
17 :: <asignment_statement> ID OP_ASIGNACION
17 :: <compound_statement> asignment_statement
17 :: <statement> compound_statement
17 :: <sequence_of_statements> statement
18 :: <case_statement_alternative> RES_WHEN var
18 :: <alternative2> alternative2 case_statement_alternative
18 :: <var> CHAR
18 :: <factor> INT
18 :: <operation_expression> factor
18 :: <math_expression> operation_expression
18 :: <expression> math_expression
18 :: <asignment_statement> ID OP_ASIGNACION
18 :: <compound_statement> asignment_statement
18 :: <statement> compound_statement
18 :: <sequence_of_statements> statement
19 :: <case_statement_alternative> RES_WHEN var
19 :: <alternative2> alternative2 case_statement_alternative
19 :: <var> CHAR
19 :: <factor> INT
19 :: <operation_expression> factor
19 :: <math_expression> operation_expression
19 :: <expression> math_expression
19 :: <asignment_statement> ID OP_ASIGNACION
19 :: <compound_statement> asignment_statement
19 :: <statement> compound_statement
19 :: <sequence_of_statements> statement
20 :: <case_statement_alternative> RES_WHEN var
20 :: <alternative2> alternative2 case_statement_alternative
20 :: <var> CHAR
20 :: <factor> INT
20 :: <operation_expression> factor
20 :: <math_expression> operation_expression
20 :: <expression> math_expression
20 :: <asignment_statement> ID OP_ASIGNACION
20 :: <compound_statement> asignment_statement
20 :: <statement> compound_statement
20 :: <sequence_of_statements> statement
21 :: <case_statement_alternative> RES_WHEN var
21 :: <alternative2> alternative2 case_statement_alternative
21 :: <var> CHAR
21 :: <factor> INT
21 :: <operation_expression> factor
21 :: <math_expression> operation_expression
21 :: <expression> math_expression
21 :: <asignment_statement> ID OP_ASIGNACION
21 :: <compound_statement> asignment_statement
21 :: <statement> compound_statement
21 :: <sequence_of_statements> statement
22 :: <case_statement_alternative> RES_WHEN var
22 :: <alternative2> alternative2 case_statement_alternative
22 :: <var> CHAR
22 :: <factor> INT
22 :: <operation_expression> factor
22 :: <math_expression> operation_expression
22 :: <expression> math_expression
```

```
22 :: <asignment_statement> ID OP_ASIGNACION
22 :: <compound_statement> asignment_statement
22 :: <statement> compound_statement
22 :: <sequence_of_statements> statement
23 :: <case_statement_alternative> RES_WHEN var
23 :: <alternative2> alternative2 case_statement_alternative
23 :: <var> CHAR
23 :: <factor> INT
23 :: <operation_expression> factor
23 :: <math_expression> operation_expression
23 :: <expression> math_expression
23 :: <asignment_statement> ID OP_ASIGNACION
23 :: <compound_statement> asignment_statement
23 :: <statement> compound_statement
23 :: <sequence_of_statements> statement
24 :: <case_statement_alternative> RES_WHEN var
24 :: <alternative2> alternative2 case_statement_alternative
24 :: <var> CHAR
24 :: <factor> INT
24 :: <operation_expression> factor
24 :: <math_expression> operation_expression
24 :: <expression> math_expression
24 :: <asignment_statement> ID OP_ASIGNACION
24 :: <compound_statement> asignment_statement
24 :: <statement> compound_statement
24 :: <sequence_of_statements> statement
25 :: <case_statement_alternative> RES_WHEN var
25 :: <alternative2> alternative2 case_statement_alternative
25 :: <var> CHAR
25 :: <factor> INT
25 :: <operation_expression> factor
25 :: <math_expression> operation_expression
25 :: <expression> math_expression
25 :: <asignment_statement> ID OP_ASIGNACION
25 :: <compound_statement> asignment_statement
25 :: <statement> compound_statement
25 :: <sequence_of_statements> statement
26 :: <case_statement_alternative> RES_WHEN var
26 :: <alternative2> alternative2 case_statement_alternative
26 :: <var> CHAR
26 :: <factor> INT
26 :: <operation_expression> factor
26 :: <math_expression> operation_expression
26 :: <expression> math_expression
26 :: <asignment_statement> ID OP_ASIGNACION
26 :: <compound_statement> asignment_statement
26 :: <statement> compound_statement
26 :: <sequence_of_statements> statement
27 :: <case_statement_alternative> RES_WHEN var
27 :: <alternative2> alternative2 case_statement_alternative
27 :: <var> CHAR
27 :: <factor> INT
27 :: <operation_expression> factor
27 :: <math_expression> operation_expression
27 :: <expression> math_expression
27 :: <asignment_statement> ID OP_ASIGNACION
27 :: <compound_statement> asignment_statement
27 :: <statement> compound_statement
27 :: <sequence_of_statements> statement
28 :: <case_statement_alternative> RES_WHEN var
28 :: <alternative2> alternative2 case_statement_alternative
28 :: <var> CHAR
28 :: <factor> INT
28 :: <operation_expression> factor
28 :: <math_expression> operation_expression
```

```
28 :: <expression> math_expression
28 :: <asignment_statement> ID OP_ASIGNACION
28 :: <compound_statement> asignment_statement
28 :: <statement> compound_statement
28 :: <sequence_of_statements> statement
29 :: <case_statement_alternative> RES_WHEN var
29 :: <alternative2> alternative2 case_statement_alternative
29 :: <var> CHAR
29 :: <factor> INT
29 :: <operation_expression> factor
29 :: <math_expression> operation_expression
29 :: <expression> math_expression
29 :: <asignment_statement> ID OP_ASIGNACION
29 :: <compound_statement> asignment_statement
29 :: <statement> compound_statement
29 :: <sequence_of_statements> statement
30 :: <case_statement_alternative> RES_WHEN var
30 :: <alternative2> alternative2 case_statement_alternative
30 :: <var> ID
30 :: <factor> INT
30 :: <operation_expression> factor
30 :: <math_expression> OP_MENOS
30 :: <expression> math_expression
30 :: <asignment_statement> ID OP_ASIGNACION
30 :: <compound_statement> asignment_statement
30 :: <statement> compound_statement
30 :: <sequence_of_statements> statement
31 :: <case_statement_alternative> RES_WHEN var
31 :: <alternative2> alternative2 case_statement_alternative
31 :: <case_statement> RES_CASE ID
31 :: <compound_statement> case_statement
31 :: <statement> compound_statement
31 :: <sequence_of_statements> statement
32 :: <program> RES_PROCEDURE
```

### 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

```
1 :: <procedure_param> empty
1 :: <declaraciones> empty
2 :: <enum_lit_spec> ID
2 :: <enum_type_def2> enum_lit_spec
2 :: <enum_lit_spec> ID
2 :: <enum_type_def2> enum_type_def2
2 :: <enum_lit_spec> ID
2 :: <enum_type_def2> enum_type_def2
2 :: <enum_lit_spec> ID
2 :: <enum_type_def2> enum_type_def2
2 :: <enum_type_def> RES_TYPE ID
2 :: <declaracion> enum_type_def
```

```
2 :: <declaraciones> declaraciones declaracion
3 :: <types> ID
3 :: <declaracion> ID SEP_DOSPUNTOS
3 :: <declaraciones> declaraciones declaracion
5 :: <factor> ID
5 :: <operation_expression> factor
5 :: <math_expression> operation_expression
5 :: <expression> math_expression
5 :: <asignment_statement> ID OP_ASIGNACION
5 :: <compound_statement> asignment_statement
5 :: <statement> compound_statement
5 :: <sequence_of_statements> statement
6 :: <factor> ID
6 :: <operation_expression> factor
6 :: <math_expression> operation_expression
6 :: <expression> math_expression
6 :: <asignment_statement> ID OP_ASIGNACION
6 :: <compound_statement> asignment_statement
6 :: <statement> compound_statement
6 :: <sequence_of_statements> sequence_of_statements statement
7 :: <factor> ID
7 :: <operation_expression> factor
7 :: <math_expression> operation_expression
7 :: <conditional_expression> math_expression
7 :: <cond_op> OP_IGUAL
7 :: <factor> ID
7 :: <operation_expression> factor
7 :: <math_expression> operation_expression
7 :: <conditional_expression> conditional_expression
7 :: <cond_expression> cond_expression
8 :: <var> STRING
8 :: <parametros> var
8 :: <procedure_call_statement> ID...
8 :: <simple_statement> procedure_call_statement
8 :: <statement> simple_statement
8 :: <sequence_of_statements> statement
9 :: <if_stat2> empty
9 :: <if_statement> RES_IF SEP_APARENTESIS
9 :: <compound_statement> if_statement
9 :: <statement> compound_statement
9 :: <sequence_of_statements> sequence_of_statements statement
10 :: <program> RES_PROCEDURE
```

## 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

```
1 :: <tipo_ent_proc> RES_IN_OUT
1 :: <types> RES_INTEGER
1 :: <procedure_param> ID SEP_DOSPUNTOS
1 :: <declaraciones> empty
3 :: <types> RES_INTEGER
```

```
3 :: <function_param> ID SEP_DOSPUNTOS
3 :: <types> RES_INTEGER
3 :: <function_param> function_param SEP_PUNTOCOMA
3 :: <types> RES_INTEGER
3 :: <declaraciones> empty
5 :: <factor> ID
5 :: <operation_expression> factor
5 :: <math_expression> operation_expression
5 :: <additive_operation> OP_MAS
5 :: <factor> ID
5 :: <operation_expression> factor
5 :: <math_expression> math_expression
5 :: <return_statement> RES_RETURN math_expression
5 :: <compound_statement> return_statement
5 :: <statement> compound_statement
5 :: <sequence_of_statements> statement
6 :: <sequenciafunction> sequence_of_statements
6 :: <subprogram_specification> RES_FUNCTION
6 :: <subprogram_declaration> subprogram_specification
6 :: <declaracion> subprogram_declaration
6 :: <declaraciones> declaraciones declaracion
9 :: <factor> INT
9 :: <operation_expression> factor
9 :: <multiplicative_operation> OP_PRODUCTO
9 :: <var> INT
9 :: <parametros> var
9 :: <var> INT
9 :: <parametros> parametros SEP_COMA
9 :: <function_call_statement> ID..
9 :: <operation_expression> operation_expression multiplicative_operation
function_call_statement
9 :: <math_expression> operation_expression
9 :: <expression> math_expression
9 :: <asignment_statement> ID OP_ASIGNACION
9 :: <compound_statement> asignment_statement
9 :: <statement> compound_statement
9 :: <sequence_of_statements> statement
10 :: <program> RES_PROCEDURE
```

## 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

```
1 :: <tipo_ent_proc> RES_IN_OUT
1 :: <types> RES_INTEGER
1 :: <procedure_param> ID SEP_DOSPUNTOS
1 :: <declaraciones> empty
3 :: <factor> ID
3 :: <operation_expression> factor
3 :: <math_expression> operation_expression
3 :: <conditional_expression> math_expression
```

```
3 :: <cond_op> OP_IGUAL
3 :: <factor> INT
3 :: <operation_expression> factor
3 :: <math_expression> operation_expression
3 :: <conditional_expression> conditional_expression
3 :: <cond_expression> cond_expression
4 :: <factor> INT
4 :: <operation_expression> factor
4 :: <math_expression> operation_expression
4 :: <expression> math_expression
4 :: <asignment_statement> ID OP_ASIGNACION
4 :: <compound_statement> asignment_statement
4 :: <statement> compound_statement
4 :: <sequence_of_statements> statement
6 :: <factor> INT
6 :: <operation_expression> factor
6 :: <math_expression> operation_expression
6 :: <expression> math_expression
6 :: <asignment_statement> ID OP_ASIGNACION
6 :: <compound_statement> asignment_statement
6 :: <statement> compound_statement
6 :: <sequence_of_statements> statement
7 :: <if_stat2> RES_ELSE sequence_of_statements
7 :: <if_statement> RES_IF SEP_APARENTESIS
7 :: <compound_statement> if_statement
7 :: <statement> compound_statement
7 :: <sequence_of_statements> statement
8 :: <program> RES_PROCEDURE
```

## Fichero Prueba-Ristra

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

## Salida Prueba-Ristra

```
2 :: <procedure_param> empty
2 :: <declaraciones> empty
3 :: <types> RES_STRING
3 :: <declaracion> ID SEP_DOSPUNTOS
3 :: <declaraciones> declaraciones declaracion
5 :: <asignment_statement> STRING
5 :: <compound_statement> asignment_statement
5 :: <statement> compound_statement
5 :: <sequence_of_statements> statement
6 :: <var> ID
6 :: <parametros> var
6 :: <procedure_call_statement> ID...
6 :: <simple_statement> procedure_call_statement
6 :: <statement> simple_statement
6 :: <sequence_of_statements> sequence_of_statements statement
7 :: <program> RES_PROCEDURE
```

## 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

```
2 :: <procedure_param> empty
2 :: <declaraciones> empty
3 :: <types> RES_INTEGER
3 :: <array_def> RES_TYPE ID
3 :: <declaracion> array_def
3 :: <declaraciones> declaraciones declaracion
4 :: <types> ID
4 :: <declaracion> ID SEP_DOSPUNTOS
4 :: <declaraciones> declaraciones declaracion
5 :: <types> RES_INTEGER
5 :: <declaracion> ID SEP_DOSPUNTOS
5 :: <declaraciones> declaraciones declaracion
7 :: <factor> INT
7 :: <operation_expression> factor
7 :: <math_expression> operation_expression
7 :: <expression> math_expression
7 :: <asignment_statement> ID OP_ASIGNACION
7 :: <compound_statement> asignment_statement
7 :: <statement> compound_statement
7 :: <sequence_of_statements> statement
8 :: <factor> ID
8 :: <operation_expression> factor
8 :: <math_expression> operation_expression
8 :: <conditional_expression> math_expression
8 :: <cond_op> OP_MENOR
8 :: <factor> INT
8 :: <operation_expression> factor
8 :: <math_expression> operation_expression
8 :: <conditional_expression> conditional_expression
8 :: <cond_expression> cond_expression
9 :: <factor> ID
9 :: <operation_expression> factor
9 :: <math_expression> operation_expression
9 :: <expression> math_expression
9 :: <factor> ID
9 :: <operation_expression> factor
9 :: <math_expression> operation_expression
9 :: <expression> math_expression
9 :: <asignment_statement> vector
9 :: <compound_statement> asignment_statement
9 :: <statement> compound_statement
9 :: <sequence_of_statements> statement
```

```
10 :: <factor> ID
10 :: <operation_expression> factor
10 :: <math_expression> operation_expression
10 :: <additive_operation> OP_MAS
10 :: <factor> INT
10 :: <operation_expression> factor
10 :: <math_expression> math_expression
10 :: <expression> math_expression
10 :: <asignment_statement> ID OP_ASIGNACION
10 :: <compound_statement> asignment_statement
10 :: <statement> compound_statement
10 :: <sequence_of_statements> sequence_of_statements statement
11 :: <while_statement> RES_WHILE SEP_APARENTESIS
11 :: <compound_statement> while_statement
11 :: <statement> compound_statement
11 :: <sequence_of_statements> sequence_of_statements statement
12 :: <factor> INT
12 :: <operation_expression> factor
12 :: <math_expression> operation_expression
12 :: <expression> math_expression
12 :: <asignment_statement> ID OP_ASIGNACION
12 :: <compound_statement> asignment_statement
12 :: <statement> compound_statement
12 :: <sequence_of_statements> sequence_of_statements statement
13 :: <factor> ID
13 :: <operation_expression> factor
13 :: <math_expression> operation_expression
13 :: <conditional_expression> math_expression
13 :: <cond_op> OP_MENOR
13 :: <factor> INT
13 :: <operation_expression> factor
13 :: <math_expression> operation_expression
13 :: <conditional_expression> conditional_expression
13 :: <cond_expression> cond_expression
14 :: <factor> ID
14 :: <operation_expression> factor
14 :: <math_expression> operation_expression
14 :: <expression> math_expression
14 :: <var> vector
14 :: <parametros> var
14 :: <procedure_call_statement> ID...
14 :: <simple_statement> procedure_call_statement
14 :: <statement> simple_statement
14 :: <sequence_of_statements> statement
15 :: <factor> ID
15 :: <operation_expression> factor
15 :: <math_expression> operation_expression
15 :: <additive_operation> OP_MAS
15 :: <factor> INT
15 :: <operation_expression> factor
15 :: <math_expression> math_expression
15 :: <expression> math_expression
15 :: <asignment_statement> ID OP_ASIGNACION
15 :: <compound_statement> asignment_statement
15 :: <statement> compound_statement
15 :: <sequence_of_statements> sequence_of_statements statement
16 :: <while_statement> RES_WHILE SEP_APARENTESIS
16 :: <compound_statement> while_statement
16 :: <statement> compound_statement
16 :: <sequence_of_statements> sequence_of_statements statement
17 :: <program> RES_PROCEDURE
```

## Fichero Prueba-While

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

## Salida Prueba-While

```
1 :: <tipo_ent_proc> RES_IN_OUT
1 :: <types> RES_INTEGER
1 :: <procedure_param> ID SEP_DOSPUNTOS
1 :: <declaraciones> empty
3 :: <factor> ID
3 :: <operation_expression> factor
3 :: <math_expression> operation_expression
3 :: <conditional_expression> math_expression
3 :: <cond_op> OP_DISTINTO
3 :: <factor> INT
3 :: <operation_expression> factor
3 :: <math_expression> operation_expression
3 :: <conditional_expression> conditional_expression
3 :: <cond_expression> cond_expression
4 :: <factor> ID
4 :: <operation_expression> factor
4 :: <math_expression> operation_expression
4 :: <additive_operation> OP_MENOS
4 :: <factor> INT
4 :: <operation_expression> factor
4 :: <math_expression> math_expression
4 :: <expression> math_expression
4 :: <asignment_statement> ID OP_ASIGNACION
4 :: <compound_statement> asignment_statement
4 :: <statement> compound_statement
4 :: <sequence_of_statements> statement
5 :: <while_statement> RES_WHILE SEP_APARENTESIS
5 :: <compound_statement> while_statement
5 :: <statement> compound_statement
5 :: <sequence_of_statements> statement
6 :: <program> RES_PROCEDURE
```

## Fichero programa

```
procedure main () is
	fichero : Fichero;
	c: Char;
	error : Integer;
	type Array_Entero is Array (1..10) of Char;
	vector : Array_Entero;
	contador : Integer;
	type enumerado is (VALOR1, VALOR2, VALOR3, VALOR4);
	enum : enumerado;
	ristra : String;
	aux : Char;

	function sumador (p1: Integer; p2: Integer) return Integer is
	begin
		return p1+p2;
	end sumador;

begin
	error := 0;
	contador := 0;
	while(contador>0) loop
		aux := vector[contador-1];
		case vector[contador-1] is
			when 'a' => enum := VALOR1;
			when 'b' => enum := VALOR2;
			when 'c' => enum := VALOR3;
			when 'd' => enum := VALOR4;
			when 'e' => enum := VALOR1;
			when 'f' => enum := VALOR2;
			when 'g' => enum := VALOR3;
			when 'h' => enum := VALOR4;
			when 'i' => enum := VALOR1;
			when 'j' => enum := VALOR2;
			when 'k' => enum := VALOR3;
			when 'l' => enum := VALOR4;
			when 'm' => enum := VALOR1;
			when 'n' => enum := VALOR2;
			when 'o' => enum := VALOR3;
			when 'p' => enum := VALOR4;
			when 'q' => enum := VALOR1;
			when 'r' => enum := VALOR2;
			when 's' => enum := VALOR3;
			when 't' => enum := VALOR4;
			when 'u' => enum := VALOR1;
			when 'v' => enum := VALOR2;
			when 'w' => enum := VALOR3;
			when 'x' => enum := VALOR4;
			when 'y' => enum := VALOR1;
			when 'z' => enum := VALOR2;
		end case;
	end loop;
	ristra := "La ristra es ";
	Put(ristra);
	Put(enum);
	ristra := "El codigo de error es ";
	Put(ristra);
	Put(error);
end main;
```

## Salida programa

```
1 :: <procedure_param> empty
1 :: <declaraciones> empty
2 :: <types> FICH
2 :: <declaracion> ID SEP_DOSPUNTOS
2 :: <declaraciones> declaraciones declaracion
3 :: <types> RES_CHAR
3 :: <declaracion> ID SEP_DOSPUNTOS
3 :: <declaraciones> declaraciones declaracion
4 :: <types> RES_INTEGER
4 :: <declaracion> ID SEP_DOSPUNTOS
4 :: <declaraciones> declaraciones declaracion
5 :: <types> RES_CHAR
5 :: <array_def> RES_TYPE ID
5 :: <declaracion> array_def
5 :: <declaraciones> declaraciones declaracion
6 :: <types> ID
6 :: <declaracion> ID SEP_DOSPUNTOS
6 :: <declaraciones> declaraciones declaracion
7 :: <types> RES_INTEGER
7 :: <declaracion> ID SEP_DOSPUNTOS
7 :: <declaraciones> declaraciones declaracion
8 :: <enum_lit_spec> ID
8 :: <enum_type_def2> enum_lit_spec
8 :: <enum_lit_spec> ID
8 :: <enum_type_def2> enum_type_def2
8 :: <enum_lit_spec> ID
8 :: <enum_type_def2> enum_type_def2
8 :: <enum_lit_spec> ID
8 :: <enum_type_def2> enum_type_def2
8 :: <enum_type_def> RES_TYPE ID
8 :: <declaracion> enum_type_def
8 :: <declaraciones> declaraciones declaracion
9 :: <types> ID
9 :: <declaracion> ID SEP_DOSPUNTOS
9 :: <declaraciones> declaraciones declaracion
10 :: <types> RES_STRING
10 :: <declaracion> ID SEP_DOSPUNTOS
10 :: <declaraciones> declaraciones declaracion
11 :: <types> RES_CHAR
11 :: <declaracion> ID SEP_DOSPUNTOS
11 :: <declaraciones> declaraciones declaracion
13 :: <types> RES_INTEGER
13 :: <function_param> ID SEP_DOSPUNTOS
13 :: <types> RES_INTEGER
13 :: <function_param> function_param SEP_PUNTOCOMA
13 :: <types> RES_INTEGER
13 :: <declaraciones> empty
15 :: <factor> ID
15 :: <operation_expression> factor
15 :: <math_expression> operation_expression
15 :: <additive_operation> OP_MAS
15 :: <factor> ID
15 :: <operation_expression> factor
15 :: <math_expression> math_expression
15 :: <return_statement> RES_RETURN math_expression
15 :: <compound_statement> return_statement
15 :: <statement> compound_statement
15 :: <sequence_of_statements> statement
16 :: <sequenciafunction> sequence_of_statements
16 :: <subprogram_specification> RES_FUNCTION
16 :: <subprogram_declaration> subprogram_specification
16 :: <declaracion> subprogram_declaration
```

```
16 :: <declaraciones> declaraciones declaracion
19 :: <factor> INT
19 :: <operation_expression> factor
19 :: <math_expression> operation_expression
19 :: <expression> math_expression
19 :: <asignment_statement> ID OP_ASIGNACION
19 :: <compound_statement> asignment_statement
19 :: <statement> compound_statement
19 :: <sequence_of_statements> statement
20 :: <factor> INT
20 :: <operation_expression> factor
20 :: <math_expression> operation_expression
20 :: <expression> math_expression
20 :: <asignment_statement> ID OP_ASIGNACION
20 :: <compound_statement> asignment_statement
20 :: <statement> compound_statement
20 :: <sequence_of_statements> sequence_of_statements statement
28 :: <factor> ID
28 :: <operation_expression> factor
28 :: <math_expression> operation_expression
28 :: <conditional_expression> math_expression
28 :: <cond_op> OP_MAYOR
28 :: <factor> INT
28 :: <operation_expression> factor
28 :: <math_expression> operation_expression
28 :: <conditional_expression> conditional_expression
28 :: <cond_expression> cond_expression
29 :: <factor> ID
29 :: <operation_expression> factor
29 :: <math_expression> operation_expression
29 :: <additive_operation> OP_MENOS
29 :: <factor> INT
29 :: <operation_expression> factor
29 :: <math_expression> math_expression
29 :: <expression> math_expression
29 :: <factor> vector
29 :: <operation_expression> factor
29 :: <math_expression> operation_expression
29 :: <expression> math_expression
29 :: <asignment_statement> ID OP_ASIGNACION
29 :: <compound_statement> asignment_statement
29 :: <statement> compound_statement
29 :: <sequence_of_statements> statement
30 :: <factor> ID
30 :: <operation_expression> factor
30 :: <math_expression> operation_expression
30 :: <additive_operation> OP_MENOS
30 :: <factor> INT
30 :: <operation_expression> factor
30 :: <math_expression> math_expression
30 :: <expression> math_expression
30 :: <factor> vector
31 :: <var> CHAR
31 :: <factor> ID
31 :: <operation_expression> factor
31 :: <math_expression> operation_expression
31 :: <expression> math_expression
31 :: <asignment_statement> ID OP_ASIGNACION
31 :: <compound_statement> asignment_statement
31 :: <statement> compound_statement
31 :: <sequence_of_statements> statement
32 :: <case_statement_alternative> RES_WHEN var
32 :: <alternative2> empty
32 :: <var> CHAR
32 :: <factor> ID
```

```
32 :: <operation_expression> factor
32 :: <math_expression> operation_expression
32 :: <expression> math_expression
32 :: <asignment_statement> ID OP_ASIGNACION
32 :: <compound_statement> asignment_statement
32 :: <statement> compound_statement
32 :: <sequence_of_statements> statement
33 :: <case_statement_alternative> RES_WHEN var
33 :: <alternative2> alternative2 case_statement_alternative
33 :: <var> CHAR
33 :: <factor> ID
33 :: <operation_expression> factor
33 :: <math_expression> operation_expression
33 :: <expression> math_expression
33 :: <asignment_statement> ID OP_ASIGNACION
33 :: <compound_statement> asignment_statement
33 :: <statement> compound_statement
33 :: <sequence_of_statements> statement
34 :: <case_statement_alternative> RES_WHEN var
34 :: <alternative2> alternative2 case_statement_alternative
34 :: <var> CHAR
34 :: <factor> ID
34 :: <operation_expression> factor
34 :: <math_expression> operation_expression
34 :: <expression> math_expression
34 :: <asignment_statement> ID OP_ASIGNACION
34 :: <compound_statement> asignment_statement
34 :: <statement> compound_statement
34 :: <sequence_of_statements> statement
35 :: <case_statement_alternative> RES_WHEN var
35 :: <alternative2> alternative2 case_statement_alternative
35 :: <var> CHAR
35 :: <factor> ID
35 :: <operation_expression> factor
35 :: <math_expression> operation_expression
35 :: <expression> math_expression
35 :: <asignment_statement> ID OP_ASIGNACION
35 :: <compound_statement> asignment_statement
35 :: <statement> compound_statement
35 :: <sequence_of_statements> statement
36 :: <case_statement_alternative> RES_WHEN var
36 :: <alternative2> alternative2 case_statement_alternative
36 :: <var> CHAR
36 :: <factor> ID
36 :: <operation_expression> factor
36 :: <math_expression> operation_expression
36 :: <expression> math_expression
36 :: <asignment_statement> ID OP_ASIGNACION
36 :: <compound_statement> asignment_statement
36 :: <statement> compound_statement
36 :: <sequence_of_statements> statement
37 :: <case_statement_alternative> RES_WHEN var
37 :: <alternative2> alternative2 case_statement_alternative
37 :: <var> CHAR
37 :: <factor> ID
37 :: <operation_expression> factor
37 :: <math_expression> operation_expression
37 :: <expression> math_expression
37 :: <asignment_statement> ID OP_ASIGNACION
37 :: <compound_statement> asignment_statement
37 :: <statement> compound_statement
37 :: <sequence_of_statements> statement
38 :: <case_statement_alternative> RES_WHEN var
38 :: <alternative2> alternative2 case_statement_alternative
38 :: <var> CHAR
```

```
38 :: <factor> ID
38 :: <operation_expression> factor
38 :: <math_expression> operation_expression
38 :: <expression> math_expression
38 :: <asignment_statement> ID OP_ASIGNACION
38 :: <compound_statement> asignment_statement
38 :: <statement> compound_statement
38 :: <sequence_of_statements> statement
39 :: <case_statement_alternative> RES_WHEN var
39 :: <alternative2> alternative2 case_statement_alternative
39 :: <var> CHAR
39 :: <factor> ID
39 :: <operation_expression> factor
39 :: <math_expression> operation_expression
39 :: <expression> math_expression
39 :: <asignment_statement> ID OP_ASIGNACION
39 :: <compound_statement> asignment_statement
39 :: <statement> compound_statement
39 :: <sequence_of_statements> statement
40 :: <case_statement_alternative> RES_WHEN var
40 :: <alternative2> alternative2 case_statement_alternative
40 :: <var> CHAR
40 :: <factor> ID
40 :: <operation_expression> factor
40 :: <math_expression> operation_expression
40 :: <expression> math_expression
40 :: <asignment_statement> ID OP_ASIGNACION
40 :: <compound_statement> asignment_statement
40 :: <statement> compound_statement
40 :: <sequence_of_statements> statement
41 :: <case_statement_alternative> RES_WHEN var
41 :: <alternative2> alternative2 case_statement_alternative
41 :: <var> CHAR
41 :: <factor> ID
41 :: <operation_expression> factor
41 :: <math_expression> operation_expression
41 :: <expression> math_expression
41 :: <asignment_statement> ID OP_ASIGNACION
41 :: <compound_statement> asignment_statement
41 :: <statement> compound_statement
41 :: <sequence_of_statements> statement
42 :: <case_statement_alternative> RES_WHEN var
42 :: <alternative2> alternative2 case_statement_alternative
42 :: <var> CHAR
42 :: <factor> ID
42 :: <operation_expression> factor
42 :: <math_expression> operation_expression
42 :: <expression> math_expression
42 :: <asignment_statement> ID OP_ASIGNACION
42 :: <compound_statement> asignment_statement
42 :: <statement> compound_statement
42 :: <sequence_of_statements> statement
43 :: <case_statement_alternative> RES_WHEN var
43 :: <alternative2> alternative2 case_statement_alternative
43 :: <var> CHAR
43 :: <factor> ID
43 :: <operation_expression> factor
43 :: <math_expression> operation_expression
43 :: <expression> math_expression
43 :: <asignment_statement> ID OP_ASIGNACION
43 :: <compound_statement> asignment_statement
43 :: <statement> compound_statement
43 :: <sequence_of_statements> statement
44 :: <case_statement_alternative> RES_WHEN var
44 :: <alternative2> alternative2 case_statement_alternative
```

```
44 :: <var> CHAR
44 :: <factor> ID
44 :: <operation_expression> factor
44 :: <math_expression> operation_expression
44 :: <expression> math_expression
44 :: <asignment_statement> ID OP_ASIGNACION
44 :: <compound_statement> asignment_statement
44 :: <statement> compound_statement
44 :: <sequence_of_statements> statement
45 :: <case_statement_alternative> RES_WHEN var
45 :: <alternative2> alternative2 case_statement_alternative
45 :: <var> CHAR
45 :: <factor> ID
45 :: <operation_expression> factor
45 :: <math_expression> operation_expression
45 :: <expression> math_expression
45 :: <asignment_statement> ID OP_ASIGNACION
45 :: <compound_statement> asignment_statement
45 :: <statement> compound_statement
45 :: <sequence_of_statements> statement
46 :: <case_statement_alternative> RES_WHEN var
46 :: <alternative2> alternative2 case_statement_alternative
46 :: <var> CHAR
46 :: <factor> ID
46 :: <operation_expression> factor
46 :: <math_expression> operation_expression
46 :: <expression> math_expression
46 :: <asignment_statement> ID OP_ASIGNACION
46 :: <compound_statement> asignment_statement
46 :: <statement> compound_statement
46 :: <sequence_of_statements> statement
47 :: <case_statement_alternative> RES_WHEN var
47 :: <alternative2> alternative2 case_statement_alternative
47 :: <var> CHAR
47 :: <factor> ID
47 :: <operation_expression> factor
47 :: <math_expression> operation_expression
47 :: <expression> math_expression
47 :: <asignment_statement> ID OP_ASIGNACION
47 :: <compound_statement> asignment_statement
47 :: <statement> compound_statement
47 :: <sequence_of_statements> statement
48 :: <case_statement_alternative> RES_WHEN var
48 :: <alternative2> alternative2 case_statement_alternative
48 :: <var> CHAR
48 :: <factor> ID
48 :: <operation_expression> factor
48 :: <math_expression> operation_expression
48 :: <expression> math_expression
48 :: <asignment_statement> ID OP_ASIGNACION
48 :: <compound_statement> asignment_statement
48 :: <statement> compound_statement
48 :: <sequence_of_statements> statement
49 :: <case_statement_alternative> RES_WHEN var
49 :: <alternative2> alternative2 case_statement_alternative
49 :: <var> CHAR
49 :: <factor> ID
49 :: <operation_expression> factor
49 :: <math_expression> operation_expression
49 :: <expression> math_expression
49 :: <asignment_statement> ID OP_ASIGNACION
49 :: <compound_statement> asignment_statement
49 :: <statement> compound_statement
49 :: <sequence_of_statements> statement
50 :: <case_statement_alternative> RES_WHEN var
```

```
50 :: <alternative2> alternative2 case_statement_alternative
50 :: <var> CHAR
50 :: <factor> ID
50 :: <operation_expression> factor
50 :: <math_expression> operation_expression
50 :: <expression> math_expression
50 :: <asignment_statement> ID OP_ASIGNACION
50 :: <compound_statement> asignment_statement
50 :: <statement> compound_statement
50 :: <sequence_of_statements> statement
51 :: <case_statement_alternative> RES_WHEN var
51 :: <alternative2> alternative2 case_statement_alternative
51 :: <var> CHAR
51 :: <factor> ID
51 :: <operation_expression> factor
51 :: <math_expression> operation_expression
51 :: <expression> math_expression
51 :: <asignment_statement> ID OP_ASIGNACION
51 :: <compound_statement> asignment_statement
51 :: <statement> compound_statement
51 :: <sequence_of_statements> statement
52 :: <case_statement_alternative> RES_WHEN var
52 :: <alternative2> alternative2 case_statement_alternative
52 :: <var> CHAR
52 :: <factor> ID
52 :: <operation_expression> factor
52 :: <math_expression> operation_expression
52 :: <expression> math_expression
52 :: <asignment_statement> ID OP_ASIGNACION
52 :: <compound_statement> asignment_statement
52 :: <statement> compound_statement
52 :: <sequence_of_statements> statement
53 :: <case_statement_alternative> RES_WHEN var
53 :: <alternative2> alternative2 case_statement_alternative
53 :: <var> CHAR
53 :: <factor> ID
53 :: <operation_expression> factor
53 :: <math_expression> operation_expression
53 :: <expression> math_expression
53 :: <asignment_statement> ID OP_ASIGNACION
53 :: <compound_statement> asignment_statement
53 :: <statement> compound_statement
53 :: <sequence_of_statements> statement
54 :: <case_statement_alternative> RES_WHEN var
54 :: <alternative2> alternative2 case_statement_alternative
54 :: <var> CHAR
54 :: <factor> ID
54 :: <operation_expression> factor
54 :: <math_expression> operation_expression
54 :: <expression> math_expression
54 :: <asignment_statement> ID OP_ASIGNACION
54 :: <compound_statement> asignment_statement
54 :: <statement> compound_statement
54 :: <sequence_of_statements> statement
55 :: <case_statement_alternative> RES_WHEN var
55 :: <alternative2> alternative2 case_statement_alternative
55 :: <var> CHAR
55 :: <factor> ID
55 :: <operation_expression> factor
55 :: <math_expression> operation_expression
55 :: <expression> math_expression
55 :: <asignment_statement> ID OP_ASIGNACION
55 :: <compound_statement> asignment_statement
55 :: <statement> compound_statement
55 :: <sequence_of_statements> statement
```

```
56 :: <case_statement_alternative> RES_WHEN var
56 :: <alternative2> alternative2 case_statement_alternative
56 :: <var> CHAR
56 :: <factor> ID
56 :: <operation_expression> factor
56 :: <math_expression> operation_expression
56 :: <expression> math_expression
56 :: <asignment_statement> ID OP_ASIGNACION
56 :: <compound_statement> asignment_statement
56 :: <statement> compound_statement
56 :: <sequence_of_statements> statement
57 :: <case_statement_alternative> RES_WHEN var
57 :: <alternative2> alternative2 case_statement_alternative
57 :: <case_statement> RES_CASE ID
57 :: <compound_statement> case_statement
57 :: <statement> compound_statement
57 :: <sequence_of_statements> sequence_of_statements statement
58 :: <while_statement> RES_WHILE SEP_APARENTESIS
58 :: <compound_statement> while_statement
58 :: <statement> compound_statement
58 :: <sequence_of_statements> sequence_of_statements statement
59 :: <asignment_statement> STRING
59 :: <compound_statement> asignment_statement
59 :: <statement> compound_statement
59 :: <sequence_of_statements> sequence_of_statements statement
60 :: <var> ID
60 :: <parametros> var
60 :: <procedure_call_statement> ID...
60 :: <simple_statement> procedure_call_statement
60 :: <statement> simple_statement
60 :: <sequence_of_statements> sequence_of_statements statement
61 :: <var> ID
61 :: <parametros> var
61 :: <procedure_call_statement> ID...
61 :: <simple_statement> procedure_call_statement
61 :: <statement> simple_statement
61 :: <sequence_of_statements> sequence_of_statements statement
62 :: <asignment_statement> STRING
62 :: <compound_statement> asignment_statement
62 :: <statement> compound_statement
62 :: <sequence_of_statements> sequence_of_statements statement
63 :: <var> ID
63 :: <parametros> var
63 :: <procedure_call_statement> ID...
63 :: <simple_statement> procedure_call_statement
63 :: <statement> simple_statement
63 :: <sequence_of_statements> sequence_of_statements statement
64 :: <var> ID
64 :: <parametros> var
64 :: <procedure_call_statement> ID...
64 :: <simple_statement> procedure_call_statement
64 :: <statement> simple_statement
64 :: <sequence_of_statements> sequence_of_statements statement
65 :: <program> RES_PROCEDURE
```

## Fichero Prueba-Fichero

```
with Text_Io.adb

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

## Salida Prueba-Fichero

```
7 :: <tipo_ent_proc> RES_IN_OUT
7 :: <types> RES_INTEGER
7 :: <procedure_param> ID SEP_DOSPUNTOS
8 :: <tipo_ent_proc> RES_IN
8 :: <types> RES_STRING
8 :: <procedure_param> procedure_param SEP_PUNTOCOMA
9 :: <tipo_ent_proc> RES_IN
9 :: <types> RES_STRING
9 :: <procedure_param> procedure_param SEP_PUNTOCOMA
11 :: <tipo_ent_proc> RES_IN
11 :: <types> RES_INTEGER
11 :: <procedure_param> ID SEP_DOSPUNTOS
13 :: <tipo_ent_proc> RES_IN
13 :: <types> RES_INTEGER
13 :: <procedure_param> ID SEP_DOSPUNTOS
14 :: <tipo_ent_proc> RES_OUT
14 :: <types> RES_STRING
14 :: <procedure_param> procedure_param SEP_PUNTOCOMA
16 :: <tipo_ent_proc> RES_IN
16 :: <types> RES_INTEGER
16 :: <procedure_param> ID SEP_DOSPUNTOS
17 :: <tipo_ent_proc> RES_IN
17 :: <types> RES_STRING
17 :: <procedure_param> procedure_param SEP_PUNTOCOMA
19 :: <types> RES_INTEGER
19 :: <function_param> ID SEP_DOSPUNTOS
19 :: <types> RES_INTEGER
23 :: <tipo_ent_proc> RES_IN
23 :: <types> RES_CHAR
23 :: <procedure_param> ID SEP_DOSPUNTOS
23 :: <tipo_ent_proc> RES_OUT
23 :: <types> RES_INTEGER
23 :: <procedure_param> procedure_param SEP_PUNTOCOMA
23 :: <declaraciones> empty
25 :: <factor> ID
26 :: <var> CHAR
26 :: <factor> INT
26 :: <operation_expression> factor
26 :: <math_expression> operation_expression
26 :: <expression> math_expression
26 :: <asignment_statement> ID OP_ASIGNACION
26 :: <compound_statement> asignment_statement
26 :: <statement> compound_statement
26 :: <sequence_of_statements> statement
27 :: <case_statement_alternative> RES_WHEN var
27 :: <alternative2> empty
27 :: <var> CHAR
27 :: <factor> INT
27 :: <operation_expression> factor
```

```
27 :: <math_expression> operation_expression
27 :: <expression> math_expression
27 :: <asignment_statement> ID OP_ASIGNACION
27 :: <compound_statement> asignment_statement
27 :: <statement> compound_statement
27 :: <sequence_of_statements> statement
28 :: <case_statement_alternative> RES_WHEN var
28 :: <alternative2> alternative2 case_statement_alternative
28 :: <case_statement> RES_CASE ID
28 :: <compound_statement> case_statement
28 :: <statement> compound_statement
28 :: <sequence_of_statements> statement
29 :: <program> RES_PROCEDURE
29 :: <program> Prototipo_func
29 :: <program> Prototipo_proc
29 :: <program> Prototipo_proc
29 :: <program> Prototipo_proc
29 :: <program> Prototipo_proc
```