```
-- alphabetical character
ALPH_C [a-zA-Z]
-- graphic char including " and Space
GC [\040-\176]
-- idem as before excluding "
GC_NON_Q [\040-\041\043-\176]

-- digit
D [0-9]
-- non zero digit
NZD [1-9]

-- numbers like x_yyy, xy_yyy, xyy_yyy, xyy(_yyy)+ where x != 0
NUM_WITH_LOWERBAR ({NZD}|{NZD}{D}|{NZD}{D}{D})(_{D}{D}{D})+
-- numbers. Zero included
NUM_WITHOUT_LOWERBAR {D}|({NZD}{D}*)

-- inner element of a string, it may be a graphic character (without the
-- double quote) or a double double quote.
STRING_BODY (\"\"|{GC_NON_Q}+)*

NUMBER {NUM_WITH_LOWERBAR}|{NUM_WITHOUT_LOWERBAR}

STRING \"{STRING_BODY}\"

IDENTIFIER {ALPH_C}(_?({ALPH_C}|{D}))*

CHARACTER \'{GC}\'

COMMENT "--"{GC}*\n

SEPARATOR [ \t\n]+
%%
"procedure"|"proc"          {rl_atom(YYLVal); RETURN Pc_procedure;}
"is"                        {rl_atom(YYLVal); RETURN Pc_is;}
"begin"                     {rl_atom(YYLVal); RETURN Pc_begin;}
"end"                       {rl_atom(YYLVal); RETURN Pc_end;}
"constant"|"const"          {rl_atom(YYLVal); RETURN Pc_const;}
"new"                       {rl_atom(YYLVal); RETURN Pc_new;}
"type"                      {rl_atom(YYLVal); RETURN Pc_type;}
"record"                    {rl_atom(YYLVal); RETURN Pc_record;}
"array"                     {rl_atom(YYLVal); RETURN Pc_array;}
"of"                        {rl_atom(YYLVal); RETURN Pc_of;}
"while"                     {rl_atom(YYLVal); RETURN Pc_while;}
"loop"                      {rl_atom(YYLVal); RETURN Pc_loop;}
"if"                        {rl_atom(YYLVal); RETURN Pc_if;}
"else"                      {rl_atom(YYLVal); RETURN Pc_else;}
"then"                      {rl_atom(YYLVal); RETURN Pc_then;}
"not"                       {rl_atom(YYLVal); RETURN Pc_not;}
"in"                        {rl_atom(YYLVal); RETURN Pc_in;}
"in out"                    {rl_atom(YYLVal); RETURN Pc_in_out;}
"null"                      {rl_atom(YYLVal); RETURN Pc_null;}
"range"                     {rl_atom(YYLVal); RETURN Pc_range;}
"and"                       {rl_atom(YYLVal); RETURN Pc_and;}
"or"                        {rl_atom(YYLVal); RETURN Pc_or;}
"("                         {rl_atom(YYLVal); RETURN Parentesi_o;}
")"                         {rl_atom(YYLVal); RETURN Parentesi_t;}
"."                         {rl_atom(YYLVal); RETURN Punt;}
","                         {rl_atom(YYLVal); RETURN Coma;}
":"                         {rl_atom(YYLVal); RETURN Dospunts;}
";"                         {rl_atom(YYLVal); RETURN Punticoma;}
":="                        {rl_atom(YYLVal); RETURN Dospuntsigual;}
"<"                         {rl_op_menor(YYLVal); RETURN Op_rel;}
">"                         {rl_op_major(YYLVal); RETURN Op_rel;}
"<="                        {rl_op_menorigual(YYLVal); RETURN Op_rel;}
">="                        {rl_op_majorigual(YYLVal); RETURN Op_rel;}
"="                         {rl_op_igual(YYLVal); RETURN Op_rel;}
"/="                        {rl_op_diferent(YYLVal); RETURN Op_rel;}
"+"                         {rl_atom(YYLVal); RETURN S_mes;}
"-"                         {rl_atom(YYLVal); RETURN S_menys;}
"*"                         {rl_atom(YYLVal); RETURN S_prod;}
"/"                         {rl_atom(YYLVal); RETURN S_quoci;}
"mod"                       {rl_atom(YYLVal); RETURN Pc_mod;}
{IDENTIFIER}                {rl_identifier(YYLVal, YYPos, yytext);  RETURN Identif;}
{NUMBER}                    {rl_literal_ent(YYLVal, YYPos, yytext); RETURN Lit;}
{STRING}                    {rl_literal_str(YYLVal, YYPos, yytext);  RETURN Lit;}
{CHARACTER}                 {rl_literal_car(YYLVal, YYPos, yytext); RETURN Lit;}
{COMMENT}                   {null;}
{SEPARATOR}                 {null;}
.                           {RETURN Error;}
```

```
%%

  with decls; use decls;
  with decls.d_arbre;

  with kobal_io; use kobal_io;
  with kobal_dfa; use kobal_dfa;
  with kobal_tokens;
  package a_lexic is
    procedure open(name: in String);
    procedure close;
    procedure YYError(s: in String);
    function YYPos return decls.d_arbre.posicio;
    function YYLex return kobal_tokens.Token;

    -- Auxiliar functions to allow external packages use these *_dfa functions
    function YYText return String;
    function YYLength return Integer;
  end a_lexic;


  with decls; use decls;
  with decls.d_arbre; use decls.d_arbre;
  with decls.d_descripcio; use decls.d_descripcio;
  with semantica; use semantica;
  with semantica.c_arbre; use semantica.c_arbre;
  with kobal_tokens; use kobal_tokens;
  package body a_lexic is

    procedure open(name: in String) is
    begin
      Open_Input(name);
    end open;

    procedure close is
    begin
      Close_Input;
    end close;

    procedure YYError(s: in String) is
    begin
      put(s);
    end YYError;

    function YYPos return decls.d_arbre.posicio is
    begin
      return (tok_begin_line, tok_begin_col);
    end YYPos;

##

    function YYText return String is
    begin
      return kobal_dfa.YYText;
    end YYText;

    function YYLength return Integer is
    begin
      return kobal_dfa.YYLength;
    end YYLength;

  end a_lexic;
```

```
-- Paraules reservades
%token Pc_procedure
%token Pc_is
%token Pc_begin
%token Pc_end
%token Pc_const
%token Pc_new
%token Pc_type
%token Pc_record
%token Pc_array
%token Pc_of
%token Pc_in
%token Pc_in_out
%token Pc_null
%token Pc_range
%token Pc_while
%token Pc_loop
%token Pc_if
%token Pc_then
%token Pc_else

-- Signes de puntuació + :=
%token Dospuntsigual
%token Dospunts
%token Coma
%token Punt Punticoma

-- Operadors
%right Pc_and Pc_or
%nonassoc Op_rel
%left S_mes S_menys
%left S_prod S_quoci Pc_mod
%token Pc_not

-- Terminals
%token Identif Lit

-- Encapsuladors
%token Parentesi_t
%token Parentesi_o

%with decls;
%with decls.d_arbre;
{
subtype YYSType is decls.d_arbre.atribut;
}

%%
PROC_PRIMA:
     PROC                                                          {rs_Root($1);}
  ;

PROC:
     Pc_procedure C_PROC Pc_is
       DECLS
     Pc_begin
        SENTS
     Pc_end Pc_procedure Punticoma                                {rs_Proc($$,$2,$4,$6);}
  ;

DECLS:
     DECLS DECL                                                    {rs_Decls($$,$1,$2);}
  |                                                                {rs_atom($$);}
  ;

DECL:
     PROC                                                          {rs_Decl($$,$1);}
  |  DECL_CONST                                                    {rs_Decl($$,$1);}
  |  DECL_VAR                                                      {rs_Decl($$,$1);}
  |  DECL_T                                                        {rs_Decl($$,$1);}
  ;

DECL_CONST:
     LID Dospunts Pc_const Identif Dospuntsigual IDX Punticoma    {rs_Decl_Const($$,$1,$4,$6);}
  ;
```

```
DECL_VAR:
     LID Dospunts Identif Punticoma                                  {rs_Decl_Var($$,$1,$3);}
  ;

DECL_T:
     Pc_type Identif Pc_is DECL_T_CONT                               {rs_Decl_T($$,$2,$4);}
  ;

DECL_T_CONT:
     Pc_new RANG Punticoma                                           {rs_Decl_T_Cont($$,$2);}
  |  Pc_record
       DCAMPS
     Pc_end Pc_record Punticoma                                      {rs_Decl_T_Cont($$,$2);}
  |  Pc_array Parentesi_o LID Parentesi_t Pc_of Identif Punticoma     {rs_Decl_T_Cont($$,$3,$6);}
  ;

DCAMPS:
     DCAMPS DCAMP                                                    {rs_DCamps($$,$1,$2);}
  |  DCAMP                                                           {rs_DCamps($$,$1);}
  ;

DCAMP:
     DECL_VAR                                                        {rs_DCamp($$,$1);}
  ;

C_PROC:
     Identif Parentesi_o ARGS Parentesi_t                            {rs_C_Proc($$,$1,$3);}
  |  Identif                                                         {rs_C_Proc($$,$1);}
  ;

ARGS:
     ARGS Punticoma ARG                                              {rs_Args($$,$1,$3);}
  |  ARG                                                             {rs_Args($$,$1);}
  ;

ARG:
     LID Dospunts MODE Identif                                       {rs_Arg($$,$1,$3,$4);}
  ;

MODE:
     Pc_in                                                           {rs_Mode_in($$);}
  |  Pc_in_out                                                       {rs_Mode_in_out($$);}
  ;

LID:
     LID Coma Identif                                                {rs_Lid($$,$1,$3);}
  |  Identif                                                         {rs_Lid($$,$1);}
  ;

RANG:
     Identif Pc_range IDX Punt Punt IDX                              {rs_Rang($$,$1,$3,$6);}
  ;

IDX:
     S_menys IDX_CONT                                                {rs_Idx_neg($$,$2);}
  |  IDX_CONT                                                        {rs_Idx_pos($$,$1);}
  ;

IDX_CONT:
     Lit                                                             {rs_Idx_Cont($$,$1);}
  |  Identif                                                         {rs_Idx_Cont($$,$1);}
  ;

SENTS:
     SENTS_NOB                                                       {rs_Sents($$,$1);}
  |  Pc_null Punticoma                                               {rs_atom($$);}
  ;

SENTS_NOB:
     SENTS_NOB SENT                                                  {rs_Sent_Nob($$,$1,$2);}
  |  SENT                                                            {rs_Sent_Nob($$,$1);}
  ;

SENT:
     S_ITER                                                          {rs_Sent($$,$1);}
  |  S_COND                                                          {rs_Sent($$,$1);}
  |  S_CRIDA                                                         {rs_Sent($$,$1);}
  |  S_ASSIGN                                                        {rs_Sent($$,$1);}
  ;
```

```
S_ITER:
     Pc_while EXPR Pc_loop
       SENTS
     Pc_end Pc_loop Punticoma                                  {rs_SIter($$,$2,$4);}
  ;

S_COND:
     Pc_if EXPR Pc_then
       SENTS
     Pc_end Pc_if Punticoma                                    {rs_SCond($$,$2,$4);}
  |  Pc_if EXPR Pc_then
       SENTS
     Pc_else
       SENTS
     Pc_end Pc_if Punticoma                                    {rs_SCond($$,$2,$4,$6);}
  ;

S_CRIDA:
     REF Punticoma                                             {rs_SCrida($$,$1);}
  ;

S_ASSIGN:
     REF Dospuntsigual EXPR Punticoma                          {rs_SAssign($$,$1,$3);}
  ;

REF:
     Identif QS                                                {rs_Ref($$,$1,$2);}
  ;

QS:
     QS Q                                                      {rs_Qs($$,$1,$2);}
  |                                                            {rs_atom($$);}
  ;

Q:
     Punt Identif                                              {rs_Q($$,$2);}
  |  Parentesi_o LEXPR Parentesi_t                             {rs_Q($$,$2);}
  ;

EXPR:
     E_AND                                                     {rs_Expr($$,$1);}
  |  E_OR                                                      {rs_Expr($$,$1);}
  |  E_OP                                                      {rs_Expr($$,$1);}
  ;

E_AND:
     E_AND Pc_and E_OP                                         {rs_EAnd($$,$1,$3);}
  |  E_OP Pc_and E_OP                                          {rs_EAnd($$,$1,$3);}
  ;

E_OR:
     E_OR Pc_or E_OP                                           {rs_EOr($$,$1,$3);}
  |  E_OP Pc_or E_OP                                           {rs_EOr($$,$1,$3);}
  ;

E_OP:
     E_OP Op_rel E_OP                                          {rs_EOpo($$,$1,$3,$2);}
  |  E_OP S_mes E_OP                                           {rs_EOps($$,$1,$3);}
  |  E_OP S_menys E_OP                                         {rs_EOpr($$,$1,$3);}
  |  E_OP S_prod E_OP                                          {rs_EOpp($$,$1,$3);}
  |  E_OP S_quoci E_OP                                         {rs_EOpq($$,$1,$3);}
  |  E_OP Pc_mod E_OP                                          {rs_EOpm($$,$1,$3);}
  |  Pc_not E_T                                                {rs_EOpnl($$,$2);}
  |  S_menys E_T                                               {rs_EOpna($$,$2);}
  |  E_T                                                       {rs_EOp($$,$1);}
  ;

E_T:
     REF                                                       {rs_ET($$,$1);}
  |  Parentesi_o EXPR Parentesi_t                              {rs_ET($$,$2);}
  |  Lit                                                       {rs_ET($$,$1);}
  ;


LEXPR:
     LEXPR Coma EXPR                                           {rs_LExpr($$,$1,$3);}
  |  EXPR                                                      {rs_LExpr($$,$1);}
  ;

%%
```

```
  package a_sintactic is
    procedure YYParse;
  end a_sintactic;


  with decls;
  with decls.d_arbre;
  with semantica; use semantica;
  with semantica.c_arbre; use semantica.c_arbre;

  with a_lexic; use a_lexic;
  with text_io; use text_io;
  with kobal_io; use kobal_io;
  with kobal_dfa; use kobal_dfa;
  with kobal_tokens; use kobal_tokens;
  with kobal_goto; use kobal_goto;
  with kobal_shift_reduce; use kobal_shift_reduce;
  package body a_sintactic is
##
  end a_sintactic;
```