```
digit [0-9]
lletra [A-Za-z]
lletdig [A-Za-z0-9]
carrep [ -~]
carrepnodoc [ -!#a-~]

%%

procedure { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_procedure;}
is        { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_is;}
begin     { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_begin;}
end       { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_end;}
in        { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_in;}
out       { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_out;}
type      { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_type;}
array     { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_array;}
constant  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_constant;}
record    { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_record;}
range     { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_range;}
of        { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_of;}
if        { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_if;}
then      { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_then;}
else      { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_else;}
for       { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_for;}
while     { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_while;}
loop      { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_loop;}
and       { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_and;}
or        { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_or;}
mod       { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return pc_mod;}

"+"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_mes;}
"-"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_menys;}
"*"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_producte;}
"/"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_divisio;}
"="  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_igual;}
":"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_dos_punts;}
"."  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_punt;}
".." { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_puntpunt;}
","  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_coma;}
";"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_punt_i_coma;}
">"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_major;}
"<"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_menor;}
">=" { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_major_igual;}
"<=" { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_menor_igual;}
"/=" { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_distint;}
":=" { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_assignacio;}
"("  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_parentesi_obert;}
")"  { rl_atom(yylval, yytext, tok_begin_line, tok_begin_col); return s_parentesi_tancat;}

{lletra}+({lletdig}*_?{lletdig})*
    { rl_id(yylval, yytext, tok_begin_line, tok_begin_col); return identificador;} --identificador

{digit}+
    { rl_lit_enter(yylval, yytext, tok_begin_line, tok_begin_col); return literal;} --digit

\"{carrepnodoc}*\"
    { rl_lit_string(yylval, yytext, tok_begin_line, tok_begin_col); return literal;} --string

\'{carrep}\'
    { rl_lit_caracter(yylval, yytext, tok_begin_line, tok_begin_col); return literal;} --caracter

"--"[^\n]*
    {null;}     --comentaris

[\t|\n|\ ]
    {null;}     --separadors

.
    {return Error;}

```

```
71
72 %%
73
74 with decls.datribut; use decls.datribut;
75 package a_lexic is
76         yylval : atribut;
77         type token is(Error, End_Of_Input, pc_and, pc_array, pc_begin, pc_constant, pc_else,
78                       pc_end, pc_for, pc_if, pc_in, pc_is, pc_loop, pc_mod, pc_new, pc_of,
79                       pc_or, pc_out, pc_procedure, pc_range, pc_record, pc_then, pc_type,
80                       pc_while, s_mes, s_menys, s_producte, s_divisio, s_igual, s_dos_punts,
81                       s_punt, s_puntpunt, s_coma, s_punt_i_coma, s_major, s_menor, s_major_igual,
82                       s_menor_igual, s_distint, s_assignacio, s_parentesi_obert,
83                       s_parentesi_tancat, literal, identificador);
84
85         function yylex return token;
86
87 end a_lexic;
88
89 package body a_lexic is
90
91 ##
92
93 end a_lexic;
94
```