```
{
exception Error
open Parser

let keywords = Hashtbl.create 17

let check_keyword s =
  try Hashtbl.find keywords s with Not_found -> IDENT s
and add_keyword s k = Hashtbl.add keywords s k

let () =
  add_keyword "var" VAR ;
  add_keyword "alloc" ALLOC ;
  add_keyword "false" (BOOL false) ;
  add_keyword "true"  (BOOL true) ;
  add_keyword "read" READ ;
  add_keyword "write" WRITE ;
  add_keyword "writeln" WRITELN ;
  add_keyword "array" ARRAY ;
  add_keyword "of" OF ;
  add_keyword "do" DO ;
  add_keyword "begin" BEGIN ;
  add_keyword "end" END ;
  add_keyword "if" IF ;
  add_keyword "then" THEN ;
  add_keyword "else" ELSE ;
  add_keyword "while" WHILE ;
  add_keyword "type" TYPE ;
  add_keyword "function" FUNCTION ;
  add_keyword "procedure" PROCEDURE ;
  add_keyword "integer" INTEGER ;
  add_keyword "boolean" BOOLEAN ;
  add_keyword "program" PROGRAM ; ()

}
rule token = parse
    [' ' '\t' '\n']
                    { token lexbuf }     (* skip blanks *)
  | ['A'-'Z''a'-'z'] + ['0'-'9'] * ''' * as lxm
                    {  check_keyword lxm }
  | ['0'-'9']+ as lxm
                    { INT(int_of_string lxm) }
  | "."             { DOT }
  | ";;"            { SEMISEMI }
  | ":="            { COLONEQUAL }
  | "<>"            { LESSGREATER }
  | "<="            { LESSEQUAL }
  | ">="            { GREATEREQUAL }
  | '<'             { LESS }
  | '>'             { GREATER }
  | ";"             { SEMI }
  | ","             { COMMA }
  | ':'             { COLON }
  | '='             { EQUAL }
  | '-'             { MINUS }
  | '+'             { PLUS }
  | '*'             { TIMES }
  | '/'             { DIV }
  | '('             { LPAREN }
  | ')'             { RPAREN }
  | '['             { LBRACKET }
  | ']'             { RBRACKET }
  | ""              { raise Error }
```