```ocaml
(* Syntaxe abstraite des expressions de type *)
type type_expr = Integer | Boolean | Array of type_expr;;
type var_list = (string * type_expr) list;;

type program = {
    global_vars : var_list;
    (* variables globales *)
    definitions : (string * definition) list;
    (* fonctions et procédures globales *)
    main : instruction list;
    (* corps principal du programme *)
  }
and definition = {
    arguments : var_list; result : type_expr option;
    (* arguments  et type du résultat *)
    local_vars : var_list;
    (* variables locales *)
    body : instruction list;
    (* corps de la fonction *)
  }

and expression =
  | Int of int | Bool of bool
    (* constantes *)
  | Bin of binop * expression * expression
    (* expressions arithétiques *)
  | Get of string
    (* accès à une variable *)
  | Function_call of string * expression list
    (* appel de fonction *)
  | Geti of expression * expression
    (* accès dans un tableau à une position « Get (e,i) <=> e[i] » *)
  | Alloc of expression * type_expr
    (* Création d'un tableau «  Alloc (e, t) <=> alloc(e : t) »
       'e' est la taille du tableau, 't' le type des éléments *)

and binop =
  | Plus | Minus | Times | Div
    (* arithmétique *)
  | Lt | Le | Gt | Ge | Eq | Ne
    (* comparaisons *)

and instruction =
  | Set of string * expression
    (* Affectation d'une variable « Set (x, e) <=> x := e » *)
  | Sequence of instruction list
    (* Suite d'instructions « Sequence [i1 ; ... ; in] <=> i1; ... ; in » *)
  | If of expression * instruction * instruction
    (* Conditionelle «  If (e, i1, i2) <=> if e then i1 else i2 » *)
  | While of expression * instruction
    (* Boucle « While (e, i) <=> while e do i *)
  | Procedure_call of string * expression list
    (* Appel de procédure *)
  | Write_int of expression
  | Writeln_int of expression
    (* Ecriture d'un entier, sans ou avec retour à la ligne *)
  | Read_int of string
    (* Lecture variable x « Read_int (x) <=> read(x) » *)
  | Seti of expression * expression * expression
    (* Affectation dans un tableau «  Seti (et, ei, e) <=> et[ei] := e » *)
```