ML2JAVA

E. Chailloux,P.Trebuchet

12 février 2015

TD

Le but de ce TD est la manipulation du compilateur `ml2java` dont voici l'arborescence :

`Java/alex.ml`	`Java/env_eval.ml`	`Java/fib.ml`	`Java/maincomp.ml`
`Java/alex.mll`	`Java/env_trans.ml`	`Java/ftest.ml`	`Java/prod.ml`
`Java/asyn.ml`	`Java/env_typeur.ml`	`Java/intereval.ml`	`Java/quest.ml`
`Java/asyn.mli`	`Java/essai.ml`	`Java/intertypeur.ml`	`Java/trans.ml`
`Java/asyn.mly`	`Java/eval.ml`	`Java/langinter.ml`	`Java/types.ml`
`Java/comp.ml`	`Java/util.ml`	`Java/lift.ml`	`Java/typeur.ml`

1 compilation d'un langage ML vers un langage impératif

Question 1

En utilisant un tableau, et un compteur entier, proposer un mecanisme qui permette, dans un langage imperatif d'implanter la spécialisation partielle de fonctions.

Question 2

Proposer un mecanisme de compilation des fonctions à la ML vers un langage à classe.

Question 3

Soit la fonction `map` suivante :

```
let rec map = function f -> function l ->
    if l =[] then []
    else (f (hd l))::: (map f (tl l));;
```

Ecrivez à la main le code produit par le compilateur suivant les schémas de compilation indiqués ci avant.
Solution:

```
/**
 *  map.java engendre par ml2java
 */


/**
 *  de'claration de la fonction map___1
 *    vue comme la classe : MLfun_map___1
 */
```

```
class MLfun_map___1 extends MLfun {

  private static int MAX = 2;

  MLfun_map___1() {super();}

  MLfun_map___1(int n) {super(n);}

  public MLvalue invoke(MLvalue MLparam){
    if (MLcounter == (MAX-1)) {
      return invoke_real(MLenv[0], MLparam);
    }
    else {
      MLfun_map___1 l = new MLfun_map___1(MLcounter+1);l.MLaddenv(MLenv,MLparam); retu
    }
  }


  MLvalue invoke_real(MLvalue f___2, MLvalue l___3) {

    {
      MLvalue T___4;
      {
        MLvalue T___5;
        MLvalue T___6;
        T___5=l___3;
        T___6=MLruntime.MLnil;
        T___4=MLruntime.MLequal( (MLlist )T___5,(MLlist )T___6);
      }
      if (((MLbool)T___4).MLaccess())
        {
          MLvalue T___7;
          T___7=MLruntime.MLnil;
          return T___7;
        }
      else
        {
          MLvalue T___8;
          {
            MLvalue T___9;
            MLvalue T___14;
            {
              MLvalue T___10;
              MLvalue T___11;
              T___10=f___2;
              {
                MLvalue T___12;
                MLvalue T___13;
                T___12=MLruntime.MLhd;
                T___13=l___3;
                T___11=((MLfun)T___12).invoke(T___13);
              }
```

```
              T___9=((MLfun)T___10).invoke(T___11);
            }
            {
              MLvalue T___15;
              MLvalue T___18;
              {
                MLvalue T___16;
                MLvalue T___17;
                T___16=map.map___1;
                T___17=f___2;
                T___15=((MLfun)T___16).invoke(T___17);
              }
              {
                MLvalue T___19;
                MLvalue T___20;
                T___19=MLruntime.MLtl;
                T___20=l___3;
                T___18=((MLfun)T___19).invoke(T___20);
              }
              T___14=((MLfun)T___15).invoke(T___18);
            }
            T___8=MLruntime.MLlist( (MLvalue )T___9,(MLlist )T___14);
          }
          return T___8;
        }
    }
  }

}
// fin de la classe MLfun_map___1
/**
 *
 */
class map {

  static MLvalue map___1= new MLfun_map___1(2);

public static void main(String []args) {

}}

// fin du fichier map.java
```

Question 4

Chaque fonction ML sera traduite par la classe MLFun Reprendre le code engendré pour la fonction `fib` et le simplifiez le à la main

```
let rec fib = function x -> if x < 2 then 1 else (fib(x-1))+(fib(x-2));;

/**
 *  fib.java engendre par ml2java
 */
```

```
/**
 *  de'claration de la fonction fib___1
 *    vue comme la classe : MLfun_fib___1
 */
class MLfun_fib___1 extends MLfun {

  private static int MAX = 1;

  MLfun_fib___1() {super();}

  MLfun_fib___1(int n) {super(n);}

  public MLvalue invoke(MLvalue MLparam){
    if (MLcounter == (MAX-1)) {
      return invoke_real(MLparam);
    }
    else {
      MLfun_fib___1 l = new MLfun_fib___1(MLcounter+1);l.MLaddenv(MLenv,MLparam); retu
    }
  }


  MLvalue invoke_real(MLvalue x___2) {

    {
      MLvalue T___3;
      {
        MLvalue T___4;
        MLvalue T___5;
        T___4=x___2;
        T___5=new MLint(2);
        T___3=MLruntime.MLltint( (MLint )T___4,(MLint )T___5);
      }
      if (((MLbool)T___3).MLaccess())
        {
          MLvalue T___6;
          T___6=new MLint(1);
          return T___6;
        }
      else
        {
          MLvalue T___7;
          {
            MLvalue T___8;
            MLvalue T___13;
            {
              MLvalue T___9;
              MLvalue T___10;
              T___9=fib.fib___1;
              {
```

```
                MLvalue T___11;
                MLvalue T___12;
                T___11=x___2;
                T___12=new MLint(1);
                T___10=MLruntime.MLsubint( (MLint )T___11,(MLint )T___12);
              }
              T___8=((MLfun)T___9).invoke(T___10);
            }
            {
              MLvalue T___14;
              MLvalue T___15;
              T___14=fib.fib___1;
              {
                MLvalue T___16;
                MLvalue T___17;
                T___16=x___2;
                T___17=new MLint(2);
                T___15=MLruntime.MLsubint( (MLint )T___16,(MLint )T___17);
              }
              T___13=((MLfun)T___14).invoke(T___15);
            }
            T___7=MLruntime.MLaddint( (MLint )T___8,(MLint )T___13);
          }
          return T___7;
        }
    }
  }

}
// fin de la classe MLfun_fib___1
/**
 *
 */
class fib {

  static MLvalue fib___1= new MLfun_fib___1(1);

public static void main(String []args) {

}}

// fin du fichier fib.java
```

Question 5
Que construit l'appel de `map fib`
Solution:
une mlfun et invoke...

Question 6
Que construit l'appel `map fib [1;2;3]`
Solution:

un invokereal.

Le point d'entrée du programme principal est défini dans le fichier `maincomp.pl` et la fonction principale utilisée dans ce fichier effectue un appel à la fonction `compile` définie dans le fichier `comp.ml`.

2 comp.ml

Voici le source du fichier `comp.ml`

```
let compile filename suffix =
  let source_name = filename ^ suffix in
  let ic = open_in_bin source_name in
  let lexbuf = Lexing.from_channel ic  in

  module_name:=filename;

  if !verbose_mode then
  begin
    print_endline "Analyse syntaxique (.)";
    print_endline "Typage (+)";
    print_endline "Traduction vers LI (*)";
    print_newline()
  end;

  let instructions = ref [] in
  try
    while true do
      let ast = parse_impl_phrase lexbuf in
      if !verbose_mode then print_string ".";
      instructions:=(translate_phrase(ast))::!instructions    done
  with
    End_of_file ->
    begin
      close_in ic;
      instructions:= List.rev (!instructions);
      prod_file filename (flat !instructions)
    end
  | x -> (close_in ic; raise x)
;;
```

Question 7
Donner la succession des étapes de compilation effectuées par ce compilateur.
Solution:

- ouverture du fichier `open_in_bin`
- initiation du Lexing `let lexbuf = Lexing.from_channel ic`
- creation des l'ast `let ast = parse_impl_phrase lexbuf`
- creation du LI `instructions:=(translate_phrase(ast))::!instructions`
- ecriture du code cible `prod_file filename (flat !instructions)`

3 prod.ml

Voici le source du fichier `prod.ml`

```
open Types;;
open Typeur;;
open Env_typeur;;
open Env_trans;;
open Langinter;;
let compiler_name = ref "ml2java";;
let object_suffix = ref ".java";;

(* des valeurs pour certains symboles de env_trans *)
pair_symbol:=",";;
cons_symbol:="::";;
ref_symbol:="ref";;

(* l'environnement initial du traducteur en liaison avec la Runtime *)
let build (s,equiv)  =
  let t =
      try List.assoc s !initial_typing_env
      with Not_found ->
        failwith ("building initial_trans_env : for symbol : "^s)
  in (s,(equiv,type_instance t));;

initial_special_env :=
 List.map build [
      "hd","MLruntime.MLhd";
      "tl","MLruntime.MLtl";
      "fst","MLruntime.MLfst";
      "snd","MLruntime.MLsnd"
];;


initial_trans_env:=

let alpha = max_unknown () in
[",",("MLruntime.MLpair", Fun_type (Pair_type (alpha,alpha),
                                    Pair_type (alpha,alpha)))]@
["::",("MLruntime.MLlist", Fun_type (Pair_type (alpha,alpha),
                                    List_type (alpha)))]@

(
List.map build
     ["true" ,"MLruntime.MLtrue";
      "false","MLruntime.MLfalse";
      "+","MLruntime.MLaddint";
      "-","MLruntime.MLsubint";
      "*","MLruntime.MLmulint";
      "/","MLruntime.MLdivint";
      "=","MLruntime.MLequal";
      "<","MLruntime.MLltint";
      "<=","MLruntime.MLleint";
      ">","MLruntime.MLgtint";
      ">=","MLruntime.MLgeint";
      "^", "MLruntime.MLconcat"

]
)
;;
(* des fonctions d'I/O *)
let output_channel = ref stdout;;
let change_output_channel oc = output_channel := oc;;
```

```
let shift_string = String.make 256 ' ';;
let out s = output_string !output_channel s;;
let out_start s nb = out ("\n"^(String.sub shift_string 0 (2*nb))^s);;
let out_end s nb = out ("\n"^(String.sub shift_string 0 nb)^"}\n");;
let out_line s = out (s^"\n");;

let out_before (fr,sd,nb) =
  if sd<>"" then out_start (sd^"=") nb
  else if fr then out_start ("return ") nb;;

let out_after  (fr,sd,nb) =
  if sd<>"" then
  begin
      out ";";
      if fr then out (("return "^sd^";"))
  end
  else if fr then out ";";;

(* des fonctions utilitaires pour commenter un peu la production *)
let header_main  s =
  List.iter out
   ["/**\n";
    " *  "^ s ^ ".java" ^ " engendre par ml2java \n";
    " */\n"]
;;

let footer_main  s =
  List.iter out
   ["// fin du fichier " ^ s ^ ".java\n"]
;;
let header_one  s =
   List.iter out
     [];;
let footer_one  s = ();;
let header_two  s =
  List.iter out
  [ "/**\n";
    " * \n";
    " */\n";
    "class "^s^" {\n"
  ]
;;
let footer_two  s = ();;
let header_three  s =
  List.iter out
  [  "\n\n";
     "public static void main(String []args) {\n"]
;;
let footer_three  s =
  List.iter out
  [ "\n}}\n\n"]
;;
(* on recuoere le  type pour une declaration precise *)
let string_of_const_type ct = match ct with
  INTTYPE    -> "MLint "
| FLOATTYPE  -> "MLdouble "
| STRINGTYPE -> "MLstring "
| BOOLTYPE   -> "MLbool "
| UNITTYPE   -> "MLunit "
```

```
;;
let rec string_of_type typ = match typ with
  CONSTTYPE t -> string_of_const_type t
| ALPHA    ->  "MLvalue "
| PAIRTYPE -> "MLpair "
| LISTTYPE -> "MLlist "
| FUNTYPE  -> "MLfun "
| REFTYPE  -> "MLref "
;;
let prod_global_var instr = match instr with
  VAR (v,t) -> out_start ("static "^"MLvalue "^(*(string_of_type t)*)v^";") 1
| FUNCTION (ns,t1,ar,(p,t2), instr) ->
    out_start ("static MLvalue "(*"fun_"^ns^" "*)^ns^"= new MLfun_"^ns^"("^(string_of_int ar)^
| _ -> ()
;;
let prod_two  ast_li =
  List.iter prod_global_var ast_li
;;
let get_param_type lv =
  List.map (function (VAR(name,typ)) -> typ
        | _ -> failwith "get_param_type" ) lv;;
let prod_const c = match c with
  INT i -> out ("new MLint("^(string_of_int i)^")")
| FLOAT f -> out ("new MLdouble("^(string_of_float f)^")")
| BOOL b  -> out ("new MLbool("^(if b then "true" else "false")^")")
| STRING s -> out ("new MLstring("^"\""^s^"\""^")")
| EMPTYLIST -> out ("MLruntime.MLnil")
| UNIT ->      out ("MLruntime.MLlrp")
;;
let rec prod_local_var (fr,sd,nb) (v,t) =
  out_start ("MLvalue "(*(string_of_type t)*)^v^";") nb;;
let rec prod_instr (fr,sd,nb) instr  = match instr with
  CONST c -> out_before (fr,sd,nb);
             prod_const c;
             out_after (fr,sd,nb)
| VAR (v,t)
          -> if (nb = 0) && ( sd = "") then ()
             else
             begin
               out_before (fr,sd,nb);
               out v;
               out_after (fr,sd,nb)
             end
| IF(i1,i2,i3) ->
              out_start "if (" nb;
              out ("((MLbool)");
              prod_instr (false,"",nb) i1 ;
              out ")";
              out".MLaccess()";
              out ")";
              prod_instr (fr,sd,nb+1) i2 ;
              out_start "else" (nb);
              prod_instr (fr,sd,nb+1) i3
| RETURN i -> prod_instr (true,"",nb) i
| AFFECT (v,i) -> prod_instr (false,v,nb) i
| BLOCK(l,i) -> out_start "{ " nb;
                  List.iter (fun (v,t,i) -> prod_local_var (false,"",nb+1)
                                              (v,t)) l;
                  List.iter (fun (v,t,i) -> prod_instr (false,v,nb+1) i) l;
```

```
                  prod_instr (fr,sd,nb+1) i;
                out_start "}" nb

| APPLY(i1,i2) ->
   out_before(fr,sd,nb);
     out ("((MLfun)");
     prod_instr (false,"",nb) i1;
     out ")";
     out ".invoke(";
     prod_instr (false,"",nb) i2;
     out")";
   out_after(fr,sd,nb)
| PRIM ((name,typ),instrl) ->
   let ltp = get_param_type instrl in
   out_before (fr,sd,nb);
   out (name^"( ("^(string_of_type (List.hd ltp))^")");
   prod_instr (false,"",nb+1) (List.hd instrl);
   List.iter2 (fun x y -> out (",("^(string_of_type y)^")");
                        prod_instr (false,"",nb+1) x)
            (List.tl instrl) (List.tl ltp);
   out ")" ;
   out_after(fr,sd,nb)

| FUNCTION _ -> ()
;;
let fun_header fn cn  =
  List.iter out
    ["\n\n";
     "/**\n";
     " *  de'claration de la fonction "^fn^"\n";
     " *    vue comme la classe : "^cn^"\n";
     " */ \n"]
;;
let prod_invoke cn  ar =
  List.iter out_line
     ["  public MLvalue invoke(MLvalue MLparam){";
      "    if (MLcounter == (MAX-1)) {"
     ];

  out "      return invoke_real(";
  for i=0 to ar-2 do
    out ("MLenv["^(string_of_int i)^"], ")
  done;
  out_line "MLparam);";

  List.iter out_line
     ["    }";
      "    else {";
      "      "^cn^" l = new "^cn^"(MLcounter+1);l.MLaddenv(MLenv,MLparam); return l;";
      "    }";
      "  }"
      ]
;;

let prod_invoke_fun cn ar t lp instr =
  out_start "MLvalue invoke_real(" 1;
  out ("MLvalue "^(List.hd lp));
  List.iter (fun x -> out (", MLvalue "^x)) (List.tl lp);
  out_line ") {";
```

```
  prod_instr (true,"",2) instr;

  out_start "}" 1;
  out_line ""
;;

let prod_fun instr = match instr with
  FUNCTION (ns,t1,ar,(lp,t2),instr) ->
      let class_name = "MLfun_"^ns in
      fun_header ns class_name ;
      out_line ("class "^class_name^" extends MLfun {");
      out_line "";
      out_line ("  private static int MAX = "^(string_of_int ar)^";") ;
      out_line "";
      out_line ("  "^class_name^"() {super();}") ;
      out_line "";
      out_line ("  "^class_name^"(int n) {super(n);}") ;
      out_line "";
      prod_invoke class_name ar;
      out_line "";
      prod_invoke_fun class_name ar t1 lp instr;
      out_line "";
      out_line "}";
      out_line ("// fin de la classe "^class_name)


|  _ -> ()
;;


let prod_one  ast_li =
  List.iter prod_fun ast_li
;;
let prod_three  ast_li =
 List.iter (prod_instr  (false,"",0) ) ast_li
;;
let prod_file filename ast_li =
  let obj_name = filename ^ !object_suffix in
  let oc = open_out obj_name in
  change_output_channel oc;
  module_name:=filename;
  try
    header_main  filename;
    header_one  filename;
    prod_one  ast_li;
    footer_one  filename;
    header_two  filename;
    prod_two  ast_li;
    footer_two  filename;
    header_three  filename;
    prod_three  ast_li;
    footer_three  filename;
    footer_main  filename;
    close_out oc
  with x -> close_out oc; raise x;;
```

Question 8

Décrire chacune trois phases de compilation : `prod_one, prod_two, prod_three`.

Solution:

—

— `prod_one` c'est la fonction qui produit les fonctions,

— `prod_two` c'est la fonction qui produit les variables globales.

— `prod_three` produit elle les instructions du `main`

4 le runtime

Voici le listing de son runtime :

```
abstract class MLvalue extends Object {

  abstract void  print();
}

class MLunit extends MLvalue
{
  private int val;

  MLunit(){val=0;}

  public void print(){System.out.print("()");}
  public int MLaccess(){return val;}
}


class MLbool extends MLvalue
{
  private boolean val;

  MLbool(boolean a){val=a;}

  public void print(){if (val) System.out.print("true");
                      else System.out.print("false");}
  public boolean MLaccess(){return val;}
}

class MLint extends MLvalue
{
   private int val;
   MLint(int a){val=a;}


   public void print(){System.out.print(val);}
   public int MLaccess(){return val;}
}

class MLdouble extends MLvalue
{
   private double val;
   MLdouble(double a){val=a;}

   public void print(){System.out.print(val);}

   public double MLaccess(){return val;}
```

```
}

class MLstring extends MLvalue
{
  private String val;
  MLstring(String a){val=a;}

  public void print(){System.out.print("\""+val+"\"");}
  public String MLaccess(){return val;}
}

class MLpair extends MLvalue
{
  private MLvalue MLfst;
  private MLvalue MLsnd;

  MLpair(MLvalue a, MLvalue b){MLfst=a; MLsnd=b;}

  public MLvalue MLaccess1(){return MLfst;}
  public MLvalue MLaccess2(){return MLsnd;}
  public void print(){System.out.print("(");
                      MLfst.print();
                      System.out.print(",");
                      MLsnd.print();
                      System.out.print(")");}

}

class MLlist extends MLvalue
{
  private MLvalue MLcar;
  private MLlist  MLcdr;

  MLlist(MLvalue a, MLvalue b){MLcar=a; MLcdr=(MLlist)b;}
  MLlist(MLvalue a, MLlist b){MLcar=a; MLcdr=b;}

  public MLvalue MLaccess1(){return MLcar;}
  public MLlist  MLaccess2(){return MLcdr;}
  public void print(){if (MLcar == null) {System.out.print("[]");}
                      else {MLcar.print();
                            System.out.print("::");
                            MLcdr.print();}}

}


abstract class MLfun extends MLvalue
{
  public int MLcounter;
  protected MLvalue[] MLenv;

  MLfun(){MLcounter=0;}
  MLfun(int n){MLcounter=0;MLenv = new MLvalue[n];}

  public  void  MLaddenv(MLvalue []O_env,MLvalue a)
   { for (int i=0; i< MLcounter; i++) {MLenv[i]=O_env[i];}
     MLenv[MLcounter]=a;MLcounter++;}

 abstract public MLvalue invoke(MLvalue x);
```

```
  public void print(){
     System.out.print("<fun> [");
     for (int i=0; i< MLcounter; i++)
           MLenv[i].print();
     System.out.print("]");
  }

}

class MLprimitive extends MLfun {

  String name="";

  MLprimitive(String n){name=n;}

  public MLvalue invoke(MLvalue l) {
    if (name.equals("hd")) return MLruntime.MLhd_real((MLlist)l);
    else if (name.equals("tl")) return MLruntime.MLtl_real((MLlist)l);
    else if (name.equals("fst")) return MLruntime.MLfst_real((MLpair)l);
    else if (name.equals("snd")) return MLruntime.MLsnd_real((MLpair)l);
    else {System.err.println("Unknown primitive "+name); return l;}
  }

}


class MLruntime {

// booleens
  public static MLbool MLtrue = new MLbool(true);
  public static MLbool MLfalse = new MLbool(false);
// unit
  public static MLunit MLlrp = new MLunit();
// nil
  public static MLlist MLnil = new MLlist(null,null);

// arithmetique sur les entiers
  public static MLint MLaddint(MLint x, MLint y) {
       return new MLint(x.MLaccess()+y.MLaccess());
  }
  public static MLint MLsubint(MLint x, MLint y) {
       return new MLint(x.MLaccess()-y.MLaccess());
  }
  public static MLint MLmulint(MLint x, MLint y) {
       return new MLint(x.MLaccess()*y.MLaccess());
  }
  public static MLint MLdivint(MLint x, MLint y) {
       return new MLint(x.MLaccess()/y.MLaccess());
  }
// fonction equal
  public static MLbool MLequal(MLvalue x, MLvalue y) {
       return new MLbool((x == y) || (x.equals(y)));
  }
// inegalites sur les entiers
  public static MLbool MLltint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()<y.MLaccess());
  }
```

```
  public static MLbool MLleint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()<=y.MLaccess());
  }

  public static MLbool MLgtint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()>y.MLaccess());
  }

  public static MLbool MLgeint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()>=y.MLaccess());
  }

// paire
  public static MLpair MLpair(MLvalue x, MLvalue y) {
        return new MLpair(x,y);
  }

// liste
  public static MLlist MLlist(MLvalue x, MLvalue y) {
        return new MLlist(x,y);
  }

//
  public static MLvalue MLconcat(MLstring x, MLstring y) {
        return new MLstring(x.MLaccess()+y.MLaccess());
  }

// acces aux champs des paires
  public static MLvalue MLfst = new MLprimitive("fst");
  public static MLvalue MLfst_real(MLpair p) {
      return p.MLaccess1();
  }

  public static MLvalue MLsnd = new MLprimitive("snd");
  public static MLvalue MLsnd_real(MLpair p) {
      return p.MLaccess2();
  }

// acces aux champs des listes
  public static MLvalue MLhd = new MLprimitive("hd");
  public static MLvalue MLhd_real(MLlist l) {
      return l.MLaccess1();
  }

  public static MLvalue MLtl = new MLprimitive("tl");
  public static MLvalue MLtl_real(MLlist l) {
      return l.MLaccess2();
  }


// la fonction d'affichage
  public static MLvalue MLprint(MLvalue x) {
     x.print();
     System.out.println();
     return MLlrp;}

}
```

Question 9

Déterminer la hiérarchie de classes correspondant à la représentation des données choisie.
Solution:

```
MLValue ->MLtout ->MLprimitive
```

Question 10

Construire directement en Java les donnés suivantes :

- `(37.2, true)`
- `''un''::''jour''::[]`
- `''un''::2::[]`
- `ref 3`
- `()`

Solution:
pair
list mlstring
erreur de typage mais bon
ref 3-¿MLint
MLpair MLUnit

Question 11

Ajouter un type tableau à cette bibliothèque pour des tableaux à la ML avec les constructeurs et accesseurs habituels. Cette question ne vise qu'à modifier le runtime !
Solution:

```
class MLArray extends MLvalue
{
    private MLvalue MLtab[];
    public MLvalue MLaccess1(MLint a)
    {
        return new MLvalue(MLtab[a.MLaccess()]);
    }
    public MLArray MLCreate(MLint a)
    {
        return new MLvalue[a.MLaccess()];
    }
    public void print(){
        for (int i=0; i< MLtab.length(); i++)
           MLtab[i].print();
    }
}
```

TME

---

5 Manipulation du runtime java

Question 12
Modifier le code du traducteur pour ne pas créer une variable temporaire uniquement affecter par une autre variable.

Question 13
Modifier le générateur de Java en détectant les expressions de mini-ML pouvant se traduire directement en Java sans passer par des variables intermédiaires.

Question 14
Que fait la méthode `invoke` pour les fonctions à un paramètre ? Modifier le traducteur pour appeler directement la méthode `invoke_real` quand tous les arguments de la fonction lui sont passés.

6 Implantation de la bibliothèque d'exécution pour C

On cherche à implanter la bibliothèque d'exécution pour C commencée en TD. Tester-la avec de petits exemples écrits directement en C à la manière des appels de la fonction map vue précédemment.

7 Génération de code C

Pour tester la bibliothèque d'exécution pour C, on cherche à modifier le générateur de code pour qu'il puisse engendrer des programmes C compatibles avec celle-ci dans le but de construire des exécutables au sens C. **Solution**:

