Produit de deux matrices d'entiers

```
# let produit = function A -> function B ->
#   let nlA = vect_length A
#   and ncA = vect_length A.(0)
#   and nlB = vect_length B
#   and ncB= vect_length B.(0) in
#     if ncA <> nlB then failwith "produit"
#     else let C = make_matrix nlA ncB 0 in
#            for i = 0 to nlA-1 do
#              for j = 0 to ncA-1 do
#                for k = 0 to ncB-1 do
#                  C.(i).(k) <- C.(i).(k)+A.(i).(j)*B.(j).(k)
#                done
#              done
#            done; C;;
produit : int vect vect -> int vect vect -> int vect vect =
<fun>
```

Itérateurs impératifs

- répèter une action sur tous les éléments d'une liste: l'itérateur prédéfini `do_list`.

```
# let rec do_list = function action -> function l ->
#   match l with
#   | [] -> ()
#   | x::l' -> action x; do_list action l';;
do_list : ('a -> 'b) -> 'a list -> unit = <fun>

# do_list print_int [1; 2; 3];;
123- : unit = ()
```

- répèter une action sur toutes les coordonnées d'un vecteur: l'itérateur prédéfini `do_vect`.

```
# let do_vect = function action -> function v ->
#   for i = 0 to vect_length v -1 do
#     action v.(i)
#   done;;
do_vect : ('a -> 'b) -> 'a vect -> unit = <fun>
# do_vect print_int [|1; 2; 3|];;
123- : unit = ()
```

- appliquer une fonction à toutes les coordonnées d'un vecteur et rendre le vecteur résultat: l'itérateur prédéfini `map_vect`.

```
# let map_vect = function f -> function v ->
#   init_vect (vect_length v) (function i -> f v.(i));;
map_vect : ('a -> 'b) -> 'a vect -> 'b vect = <fun>
# map_vect (function i -> i*i) [|1; 2; 3|];;
- : int vect = [|1; 4; 9|]
```