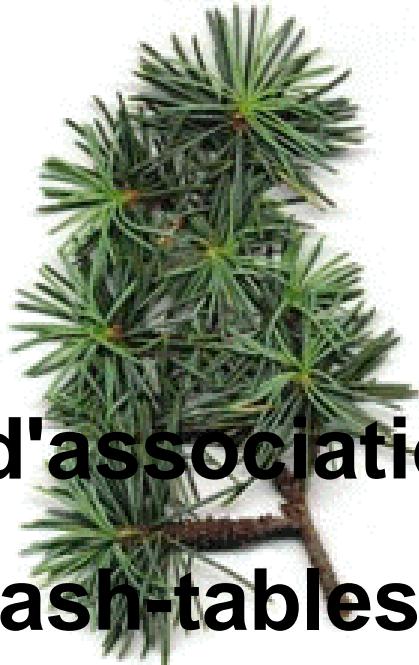


<https://laurentbloch.org/BlogLB/Liste-d-associations-ou-hash>



Liste d'associations ou hash-tables ?

- Zinformatiques - Cours de bioinformatique au CNAM -

Date de mise en ligne : samedi 14 janvier 2006

Copyright © Blog de Laurent Bloch - Tous droits réservés

Ce texte reprend le cours du 12 janvier, ainsi que le thème d'un [article précédent](#) sur le même sujet. Il s'agit d'écrire un programme pour gérer une collection d'observations météorologiques, que l'utilisateur doit saisir au fur et à mesure qu'elles lui parviennent, en nombre indéterminé. Il faut aussi pouvoir interroger cette collection d'observations, tant qu'elle est en mémoire : la solution suggérée est d'utiliser une *fermeture*, qui encapsulera le programme, avec lequel on communiquera par passage de messages, et la structure de données qui contiendra les observations.

La première solution proposée repose sur une liste d'associations :

```
(define (make-meteo)
  (let ((une-meteo '()))
    (lambda message
      (case (car message)
        ((peupler)
         (let boucle ()
           (display "Entrez ville et température : ")
           (let* ((ville (read))
                  (temp (read)))
             (if (string=? ville "")
                 #f
                 (begin
                   (set! une-meteo
                         (cons (cons ville temp)
                               une-meteo))
                   (boucle))))))
        ((interroger)
         (let* ((ville (cadr message))
                (reponse (assoc ville une-meteo)))
           (if reponse
               (cdr reponse)
               "La ville demandée n'a pas d'observation")))
        ((afficher)
         (print une-meteo))))))
```

Une seconde solution, plus efficace, utilise les *hash-tables* :

Liste d'associations ou hash-tables ?

```
(define (make-meteo)
  (let* ((n 23) ; nombre d'éléments du vecteur
         (une-meteo (make-vector n '())))
    (lambda message
      (case (car message)
        ((peupler)
         (let boucle ()
           (display "Entrez ville et température : ")
           (let* ((la-ville (read))
                  (numero (read)))
             (if (not (string=? la-ville ""))
                 (begin
                   (vec:ajouter! la-ville
                     numero
                     une-meteo)
                   (boucle))))))
        ((interroger)
         (let* ((la-ville (cadr message))
                (la-case (hash la-ville n))
                (reponse
                  (assoc la-ville (vector-ref une-meteo la-case))))
           (if reponse
               (cdr reponse)
               "La ville demandée n'observe pas sa température")))
        ((donner)
         une-meteo)
        ((afficher)
         (vector:for-each print une-meteo))))))
```

Liste d'associations ou hash-tables ?

Il faut encore quelques procédures auxiliaires :

```
(define (hash nom taille-table)
  (remainder
    (apply + (map char->integer (string->list nom)))
    taille-table))

(define (assoc:ajouter! nom numero liste)
  (cons (cons nom numero) liste))

(define (vec:ajouter! nom numero vecteur)
  (let* ((n (vector-length vecteur))
         (une-case (hash nom n)))
    (vector-set! vecteur une-case
      (assoc:ajouter!
        nom
        numero
        (vector-ref vecteur une-case)))))

(define (vector:for-each proc V)
  (let ((longueur (vector-length V)))
    (let boucle ((index 0))
      (if (not (= index longueur))
          (begin
            (proc (vector-ref V index))
            (boucle (+ index 1)))))))
```