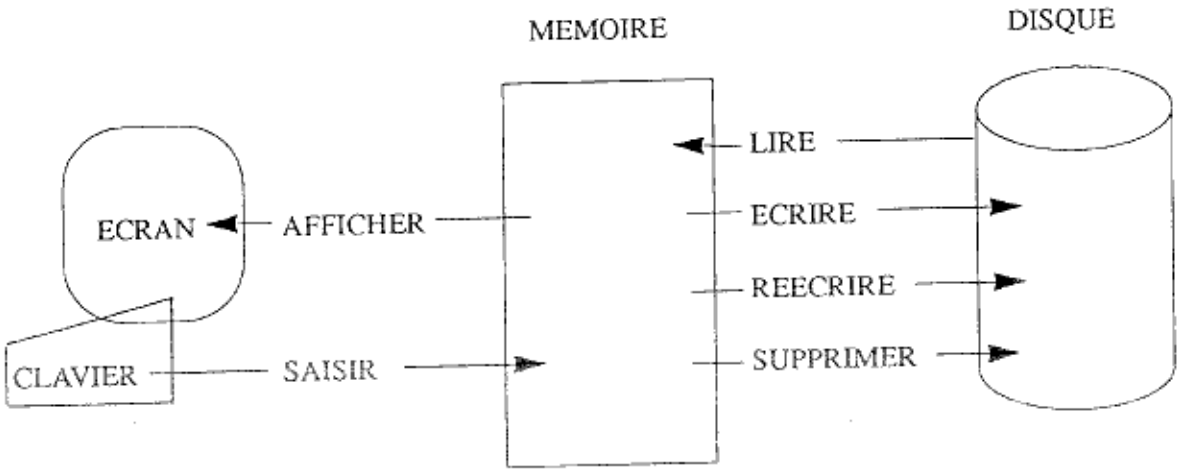
5

Les fichiers

I – Introduction

Toutes les structures de données que nous avons utilisées jusqu'à maintenant étaient stockées en mémoire, c'est-à-dire dans une zone volatile de la machine. En mémoire, la durée de vie d'une variable est égale au temps d'exécution du programme. Nous comprenons facilement que cette solution n'est pas envisageable pour l'informatisation d'une entreprise. Les données concernant un employé, un produit, un fournisseur ou un contrat doivent être mémorisées plus longtemps que la durée d'un programme. Les valeurs traitées doivent être recopiées sur un support non volatile, un disque, une disquette, une bande, une cassette ...

Nous connaissons les instructions pour échanger des valeurs entre l'utilisateur et la machine, à l'aide du clavier et de l'écran : SAISIR et AFFICHER, découvrons celles d'échanges entre l'ordinateur et le support non volatile d'informations : LIRE, ECRIRE, REECRIRE, SUPPRIMER, ...



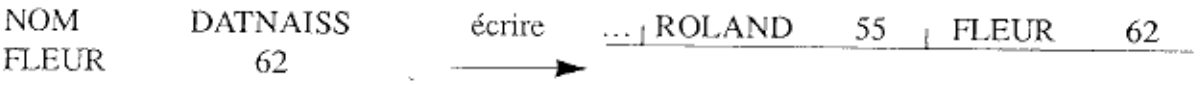
Exemple 1

Supposons que nous avons stocké les noms et les années de naissance des employés d'une entreprise :

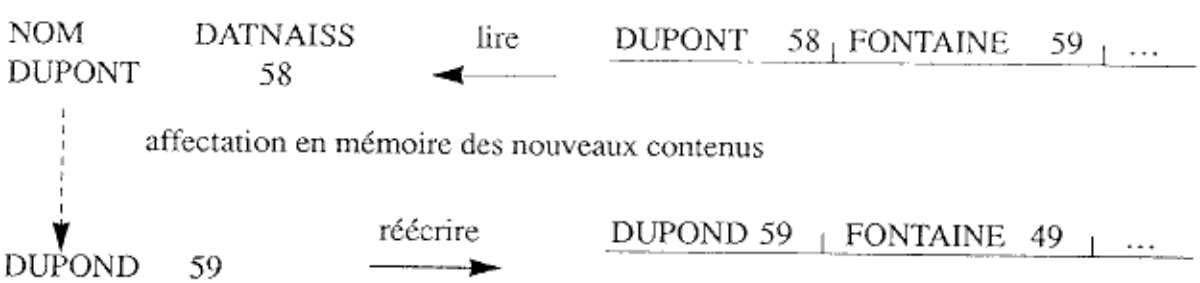
DUPONT 58 | FONTAINE 49 | DURAND 61 | ABAR 38 | ROLAND 55 |

Une instruction de lecture, LIRE EMPLOYE, copie du disque vers la mémoire les informations d'UN employé. Deux variables déclarées en mémoire d'identifiants NOM et DATNAIS reçoivent les informations lues.

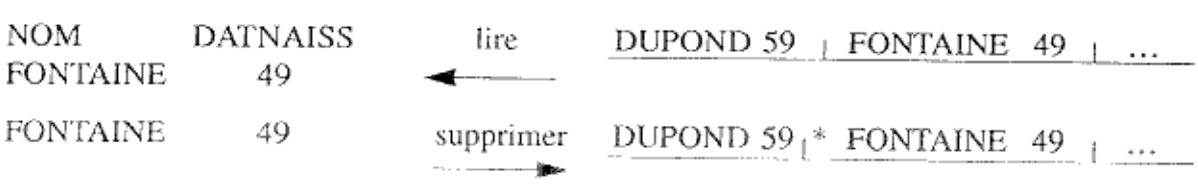
Une instruction d'écriture, ECRIRE EMPLOYE, copie les informations en mémoire sur une place libre du support :



Une instruction de réécriture, REECRIRE EMPLOYE, recopie les informations en mémoire à la place des dernières informations lues. Après l'instruction de lecture amenant les informations de l'employé "DUPONT" en mémoire, si nous ajoutons "1" à sa date de naissance et corrigeons l'orthographe du nom en remplaçant le "T" par un "D", nous obtenons :



Une instruction de suppression, SUPPRIMER EMPLOYE, met à jour un indicateur devant les dernières informations lues, pour indiquer que, logiquement, ces valeurs ne sont plus traitées. Nous parlons de suppression logique et non physique. La suppression physique nécessite d'effectuer un décalage de toutes les informations vers la gauche, mais ce traitement risque d'être très long ! Après la lecture des données concernant l'employé "FONTAINE", une instruction de suppression provoque ceci :



(le caractère "*" est supposé être l'indicateur de suppression)

DÉFINITION

Nous nommons ENREGISTREMENT (article ou enregistrement logique) la collection d'informations mémorisées pour définir UN des objets traités.

Nous appelons FICHIER la collection des enregistrements.

```
FICHIER   FIC-EMPLOYE
ENREGISTREMENT  rubriques :  nom-employé, date-de-naissance
                    valeurs :  DUPOND,59
                               FONTAINE,49
                               DURAND ......
```

II – Mise à jour d'un fichier d'organisation séquentielle

DÉFINITION

Une première solution pour implanter les enregistrements sur le support est l'organisation séquentielle. Les enregistrements sont écrits les uns à la suite des autres, la lecture commence toujours par le premier, puis le second, son suivant et son suivant jusqu'au dernier.

DÉCLARATION ET INSTRUCTIONS

Nous déclarons un fichier ainsi :

```
FICHIER <nom-fichier> (ORGANISATION SEQUENTIELLE)
        ENREGISTREMENT <nom-enregistrement-fichier>
             <rubrique-1> : <type>
             <rubrique-2> : <type>
             <rubrique-3> : <type>
             ....
        FIN-ENREGISTREMENT
```

La première instruction du programme est l'ouverture du fichier, elle indique si le fichier va être lu (L) ou écrit (E), ou lu, écrit et mis à jour (L/E).

```
OUVRIR   <nom-fichier>   (L)
                         (E)
                         (L/E)
```

La dernière instruction du programme est la fermeture du fichier.

```
FERMER   <nom-fichier>
```

Les autres instructions ne concernent pas globalement le fichier mais un de ses enregistrement, nous les notons :

```
ECRIRE <nom-enregistrement-fichier>

LIRE <nom-enregistrement-fichier>
SI FIN FICHIER
ALORS <traitement-1>
SINON <traitement-2>
FSI
```

La lecture nécessite un test, car les lectures séquentielles se terminent lorsque tous les enregistrements ont été lus un à un. Un indicateur booléen du système de gestion de fichier nous permet d'identifier le fait qu'il n'y a plus d'enregistrement à lire, nous le nommons FIN-FICHIER.

```
REECRIRE nom-enregistrement-fichier

SUPPRIMER nom-enregistrement-fichier
```

Les deux dernières instructions ne peuvent être exécutées qu'après la lecture d'un enregistrement. C'est l'enregistrement le plus recemment lu qui est concerné par l'opération de réécriture ou de suppression. Seuls les supports adressables (disques et disquettes) permettent l'exécution de ces deux derniers ordres.

■ Algorithme – exercice 1

CREATION d'un fichier séquentiel sur disque des employés d'une entreprise, contenant le nom, le prénom, la date de naissance et la fonction de l'employé.

MÉTHODE

– Déclaration du fichier employé.
– Saisie des valeurs et stockage en mémoire.
– Validation des informations saisies.
– Ecriture des données de la mémoire sur le support.
– Question à l'utilisateur pour recommencer ou non le traitement de saisie.

```
programme création
fichier employé (organisation séquentielle)
   enregistrement enr-employé
     e-nom        :  caractères
     e-prénom     :  caractères
     e-datnais    :  entier
     e-fonction   :  caractères
   fin-enregistrement
     (* notons la présence du préfixe e- pour différencier les rubriques
     des fichiers et les variables mémoires. La lettre "e" est simplement
     l'initiale du nom du fichier *)
var réponse, valid : caractères
```

```
début
     (* ouverture du fichier *)
   ouvrir employé (E)
   répéter
     afficher "donner le nom et le prénom"
     saisir e-nom, e-prénom
     afficher "donner la date de naissance"
     saisir e-datnais
     afficher "donner la fonction de l'employé"
     saisir e-fonction
     afficher "valider vous les informations saisies (o/n) ?"
     saisir valid
     si  valid = "o"
     alors (* écriture sur disque *)
           écrire enr-employé
     fsi
         (* question à l'utilisateur *)
     afficher "autre employé à saisir (o/n) ?"
     saisir réponse
   jqa réponse = "n"
         (* fermeture du fichier *)
   fermer employé
fin
```

■ Algorithme – exercice 2

AFFICHAGE sur l'écran de toutes les informations contenues dans le fichier.

MÉTHODE

Nous prenons l'habitude d'effectuer une première lecture du fichier, pour différencier le cas d'un fichier vide (fin de fichier dès la première lecture), et le cas de la fin du traitement des enregistrements d'un fichier (fin de fichier rencontrée dans une boucle).
– Première lecture.
– Boucle de lecture séquentielle et d'affichage (nous faisons attention à traiter le premier enregistrement lu, avant d'effectuer une nouvelle lecture).

```
programme affiche
fichier employé (organisation séquentielle)
   enregistrement enr-employé
     e-nom       :  caractères
     e-prénom    :  caractères
     e-datnais   :  entier
     e-fonction  :  caractères
   fin-enregistrement
début
     (* ouverture du fichier *)
   ouvrir employé (L)
     (* première lecture *)
   lire enr-employé
   si fin-fichier
```

```
    alors afficher "le fichier des employés est vide".
    sinon afficher "affichage du fichier des employés"
          afficher " nom prénom date.naiss. fonction "
          répéter
               afficher e-nom, e-prénom, e-datnaiss, e-fonction
               lire enr-employé
          jqa fin fichier
    fsi
    fermer employé
fin
```

■ Algorithme – exercice 3

CONSULTATION des informations stockées sur un employé de l'entreprise, nous le sélectionnons par son nom.

MÉTHODE

– Saisie du nom recherché.
– Lecture du premier enregistrement.
– Lecture du suivant jusqu'à ce que la recherche aboutisse ou échoue.
– Affichage des informations à consulter.

```
programme consultation
fichier employé (organisation séquentielle)
   enregistrement enr-employé
     e-nom       : caractères
     e-prénom    : caractères
     e-datnais   : entier
     e-fonction  : caractères
   fin-enregistrement
var nom : caractères
   trouvé : booléen
début
     (* ouverture du fichier *)
   ouvrir employé (L)
   afficher "donner le nom recherché"
   saisir nom
     (* première lecture *)
   lire enr-employé
   si    fin-fichier
   alors afficher "le fichier des employés est vide"
   sinon (* recherche de l'enregistrement de l'employé *)
     recherche (employé, nom, trouvé)
               (* affichage du résultat de la recherche *)
     si non trouvé
     alors afficher nom, " n'est pas dans cette entreprise"
     sinon afficher e-nom, e-prénom, " né le ", e-datnaiss
           afficher "travaille dans cette entreprise "
           afficher "avec la fonction de ", e-fonction
     fsi
   fsi
     (* fermeture du fichier *)
   fermer employé
fin
```

```
procédure recherche   (données : employé, élément
                      résultat : trouvé)
var fini : booléen
début
   fini ← .faux.
   trouvé ← e-nom = élément
     (* recherche séquentielle *)
   tant que non trouvé et non fini
     lire enr-employé
     si fin-fichier
     alors fini ← .vrai.
     sinon trouvé ← e-nom = élément
     fsi
   ftq
fin
```

REMARQUE

Nous devons utiliser une boucle itérative et non répétitive, car le premier enregistrement déjà lu peut être celui recherché. Ceci nous oblige à utiliser un booléen (fini) pour repérer la fin de fichier, car le booléen du système (fin-fichier) ne peut être testé qu'une seule fois et juste après l'instruction de lecture. Nous utiliserons le booléen "fini" de nombreuses fois par la suite.

■ Algorithme – exercice 4

MODIFICATION de la fonction d'un employé, désigné par son nom.

MÉTHODE

– Saisie du nom de l'employé concerné par la modification.

– Recherche de l'enregistrement de cet employé dans le fichier (ce traitement est identique à celui effectué pour la consultation).

– Saisie de la nouvelle fonction.

– Mise à jour de l'enregistrement du disque.

```
programme modification
fichier employé (organisation séquentielle)
   enregistrement enr-employé
     e-nom       : caractères
     e-prénom    : caractères
     e-datnais   : entier
     e-fonction  : caractères
   fin-enregistrement
var nom : caractères
   trouvé : booléen
début
   ouvrir employé (L/E)
   afficher "donner le nom de l'employé qui change de fonction "
   saisir nom
     (* recherche *)
   recherche (employé, nom, trouvé)
     (* traitement *)
   si   trouvé
```

```
    alors afficher "modification de ", e-nom, e-prénom
         afficher "fonction de ", e-fonction
         afficher "donner la nouvelle fonction "
         saisir e-fonction
              (* réécriture sur le disque *)
         réécrire enr-employé
    sinon afficher nom, " ne travaille pas dans cette entreprise !"
    fsi
    fermer employé
fin
```

■ Algorithme – exercice 5

SUPPRESSION d'un employé de l'entreprise.

MÉTHODE

Nous prenons l'habitude de demander une validation à l'utilisateur avant d'exécuter l'instruction de suppression logique dans le fichier. Un enregistrement peut contenir des informations très importantes et difficiles à reconstituer, et leur suppression par erreur n'est pas toujours réparable.

– Saisie du nom de l'employé qui quitte l'entreprise.

– Recherche de l'enregistrement concernant cet employé dans le fichier (ce traitement est identique à celui effectué pour la consultation ou la modification).

– Affichage de l'enregistrement et saisie de la validation de la suppression.

– Suppression logique de l'enregistrement.

```
programme suppression
fichier employé (organisation séquentielle)
   enregistrement enr-employé
     e-nom       : caractères
     e-prénom    : caractères
     e-datnais   : entier
     e-fonction  : caractères
   fin-enregistrement
var nom, réponse : caractères
   trouvé : booléen
début
   ouvrir employé (L/E)
   afficher "donner le nom de l'employé qui part "
   saisir nom
     (* recherche *)
   recherche (employé, nom, trouvé)
     (* traitement *)
   si trouvé
   alors (* demande de validation *)
     afficher "validez-vous la suppression de ", e-nom, e-prénom
     afficher "né le ", e-datnaiss
     afficher "remplissant la fonction de ", e-fonction, " (O/N) ?"
     saisir réponse
```

```
si réponse = "O"
alors    (* suppression sur disque *)
   supprimer enr-employé
fsi
sinon afficher nom, " ne travaille pas dans cette entreprise"
fsi
fermer employé
fin.
```

III - Edition sur imprimante

1 - ÉDITION SIMPLE

Nous différençons l'affichage à l'écran des informations d'un fichier et l'édition à l'imprimante de ces mêmes données. Nous constatons que chaque utilisateur possède toujours un écran individuel, mais rarement une imprimante personnelle, il ne peut pas imprimer quand il le désire. Nous effectuons des IMPRESSIONS DIFFEREES en constituant une copie de l'état désiré dans un fichier stocké sur le disque, et en demandant par la suite au système d'exploitation d'éditer ce fichier intermédiaire. L'imprimante n'est pas retardée par les calculs contenus dans le programme, ni par le temps de recherche des enregistrements, elle ne risque pas d'être mise hors service par un programme qui se termine de façon anormale.

Le fichier intermédiaire doit être déclaré, il est toujours organisé en séquentiel, ses enregistrements sont de la taille du papier d'impression (40, 80 ou 132 caractères). Nous différençons plusieurs types d'enregistrement : un ou plusieurs d'en-tête, un ou plusieurs de détail, un ou plusieurs de totaux. Nous numérotons séquentiellement chaque ligne distincte en la préfixant par le mot "ligne".

Exemple :

Etat d'édition des employés de la société. Nous remarquons l'importance de dater tous les

```
        fsi
        liremaj (mise-à-jour)
      ftq (* cas de m-nom = HV compris *)
      si non supp
      alors écrire enr-empmaj
      fsi
fin
```

V – Mise à jour d'un fichier d'organisation séquentielle indexée

DÉFINITION

Une seconde solution pour implanter les enregistrements sur le support est l'organisation séquentielle indexée. Les enregistrements sont rangés en respectant l'ordre croissant d'une des rubriques, appelée "clé", qui permet de repérer de façon unique un enregistrement. Le système de gestion de fichier gère les correspondances entre la place d'un enregistrement sur le support et la valeur de sa clé dans une table nommée TABLE D'INDEX.

Les seuls supports aptes à recevoir un fichier organisé en séquentiel indexé sont les SUPPORTS ADRESSABLES (disques, disquettes), l'adresse est la place de l'enregistrement.

La clé peut être soit une rubrique, soit un groupe de rubriques du fichier. Son choix doit avant tout assurer l'unicité d'un enregistrement pour une valeur de clé donnée.

L'organisation séquentielle indexée permet un ACCES DIRECT à un enregistrement précis, grâce à la table d'index mise à jour par le système de gestion de fichier, ou un ACCES SEQUENTIEL, similaire à celui effectué dans un fichier organisé en séquentiel.

DÉCLARATION ET INSTRUCTIONS

```
FICHIER <nom-fichier> (ORGANISATION INDEXEE ACCES SEQUENTIEL
                                            DIRECT
                       CLE = rubrique-x)
    ENREGISTREMENT <nom-enregistrement-fichier>
        <rubrique-1> : <type>
        <rubrique-2> : <type>
        .....
    FIN-ENREGISTREMENT
```

Instruction d'ouverture et de fermeture identiques à celles définies pour l'organisation séquentielle.

```
        (* ACCES DIRECT *)
    (* transfert de clé *)
```

```
LIRE <nom-enregistrement-fichier>
SI ERREUR-CLE
ALORS <traitement-1>  (* enregistrement inexistant *)
SINON <traitement-2>
FSI
```

(* ACCES SEQUENTIEL *)

```
LIRE <nom-enregistrement-fichier>
SI FIN-FICHIER
ALORS <traitement-1>
SINON <traitement-2>
FSI

ECRIRE <nom-enregistrement-fichier>
SI ERREUR-CLE
ALORS <traitement-1> (* clé en double *)
SINON <traitement-2>
FSI

REECRIRE <nom-enregistrement-fichier>
SI ERREUR-CLE
ALORS <traitement-1> (* problème réécriture sur le disque, clé modifiée *)
SINON <traitement-2>
FSI

SUPPRIMER <nom-enregistrement-fichier>
SI ERREUR-CLE
ALORS <traitement-1> (* problème réécriture booléen système,
                        suppression logique *)
SINON <traitement-2>
FSI
```

ERREUR-CLE est un booléen du système de gestion de fichier. Le test de son contenu doit être effectué après toutes les actions exécutées sur le fichier, sans exception.

■ Algorithme – exercice 13

CREATION d'un fichier sur disque des produits en vente dans un magasin de peinture. Chaque produit possède un numéro qui l'identifie de façon unique, une désignation, une quantité en stock, un prix unitaire.

MÉTHODE

La méthode est identique à celle de création d'un fichier séquentiel.

- Déclaration du fichier produits.
- Saisie des valeurs.
- Validation de la saisie.
- Ecriture des informations de la mémoire sur le support.
- Question et reprise conditionnelle du traitement.

```
programme création
fichier produits(organisation indexée accès direct
                 clé = p-num)
    enregistrement enr-produits
        p-num      :  entier
        p-design   :  caractères
        p-stock    :  entier
        p-prix     :  entier
    fin-enregistrement
var  réponse, valid : caracteres
début
   ouvrir produits (E)
   répéter
    afficher "donner le numéro de produit"
    saisir p-num
    afficher "donner la désignation"
    saisir p-design
    afficher "donner la quantité en stock"
    saisir p-stock
    afficher "donner le prix unitaire"
    saisir p-prix
        (* validation *)
    afficher "validez-vous les informations saisies (o/n) ?"
    saisir valid
    si  valid = "o"
    alors    (* écriture sur disque *)
        écrire enr-produits
        si erreur-clé
        alors afficher "écrire impossible, clé en double "
        fsi
    fsi
    afficher "autre produit à saisir (o/n) ?"
    saisir réponse
   jqa réponse = "n"
   fermer produits
fin
```

REMARQUE

Plusieurs systèmes de gestion de fichiers imposent à l'utilisateur de saisir les enregistrements dans l'ordre croissant des clés, lors de la création d'un fichier organisé en séquentiel indexé. Nous obtenons dans ce cas, l'affichage d'un message système dès qu'un enregistrement a une clé inférieure à la dernière valeur de clé traitée.

■ Algorithme – exercice 14

AFFICHAGE sur l'écran de toutes les informations contenues dans le fichier. Ce traitement nécessite un accès séquentiel, l'algorithme est donc identique à l'affichage des données d'un fichier séquentiel.

```
programme affichage
fichier produits(organisation indexée accès séquentiel
                 clé = p-num)
    enregistrement enr-produits
        .....
    fin-enregistrement
début
   ouvrir produits (L)
   lire enr-produits
   si    fin-fichier
   alors afficher "le fichier des produits est vide"
   sinon afficher " fichier des produits    "
         afficher "numéro     désignation   stock  prix"
         répéter
              afficher p-num, p-design, p-stock, p-prix
              lire enr-produits
         jqa fin-fichier
   fsi
   fermer produits
fin
```

■ Algorithme – exercice 15

CONSULTATION des informations stockées sur un produit du fichier. Ce traitement est beaucoup plus rapide avec cette organisation, qui permet l'accès direct.

MÉTHODE
- Saisie du numéro de produit recherché.
- Lecture directe de cet enregistrement.
- Affichage des informations à consulter.

```
programme consultation
fichier produits (organisation indexée accès direct
                 clé = p-num)
    enregistrement enr-produits
        p-num      :  entier
        p-design   :  caractères
        p-stock    :  entier
        p-prix     :  entier
    fin-enregistrement
var numéro : entier
début
   ouvrir produits (L)
   afficher "donner le numéro du produit recherché"
   saisir numéro
     (* transfert valeur de la clé et lecture *)
   p-num ← numéro
   lire enr-produits
   si erreur-clé
   alors afficher "le produit n° ",numéro," n'existe pas"
```

```
    sinon afficher "produit n°          ",p-num
          afficher "désignation         ",p-design
          afficher "quantité en stock ",p-stock
          afficher "prix unitaire       ",p-prix
    fsi
    fermer produits
fin
```

■ Algorithme – exercice 16

MODIFICATION du prix unitaire d'un produit du fichier. Ce traitement utilise la même méthode que pour la modification d'un enregistrement d'un fichier séquentiel.

Deux différences :
– la procédure de recherche n'utilise pas d'itérative (elle est plus rapide d'exécution, grâce à l'accès direct);
– un test est nécessaire après l'instruction de réécriture.

```
programme modification
fichier produits (organisation indexée accès direct
                  clé = p-num)
     (* même déclaration que précédemment *)
var numéro : entier
   trouvé : booléen
début
   ouvrir produits (L/E)
   afficher "donner le numéro du produit qui change de prix "
   saisir numéro
   recherche (produits, numéro, trouvé)
   si trouvé
   alors afficher "modification du produit ",p-num,p-design
         afficher "ancien prix ",p-prix, "nouveau prix ? "
         saisir p-prix
         réécrire enr-produits
         si erreur-clé
         alors afficher "problème de réécriture sur disque ! "
         fsi
    sinon afficher "numéro : ",numéro," non trouvé dans le fichier"
    fsi
    fermer produits
fin

procédure recherche
   (données : produits, numéro
   résultat : trouvé)
début
   p-num ← numéro
```

La SUPPRESSION d'un enregistrement du fichier séquentiel indexé utilise le très simple sous-programme de recherche qui précède. Le programme principal, identique à celui de suppression dans un fichier séquentiel, effectue un test d'erreur de clé après la réécriture.

Les problèmes d'EDITIONS simples ou avec ruptures et d'APPAREILLAGES manipulant un fichier d'organisation séquentielle indexée utilisent l'accès séquentiel. Les algorithmes vus aux paragraphes III et IV de ce chapitre sont donc applicables à tous les fichiers, quelles que soient leurs organisations.

VI – Mise à jour d'un fichier d'organisation relative

DÉFINITION

Une troisième solution pour implanter les enregistrements sur le support est l'organisation directe. Elle gère l'emplacement d'un enregistrement, grâce à un calcul d'adresse à partir d'une valeur numérique de clé. La difficulté est d'obtenir une règle de calcul qui permette de gérer au mieux la place réservée sur le support, c'est-à-dire qu'un grand pourcentage de place doit être occupé et qu'un très faible pourcentage de valeurs de clé doit aboutir au même résultat d'adresse. Ce dernier cas est tout de même géré par le système de gestion de fichiers qui implante les enregistrements dont la valeur calculée d'adresse est déjà occupée, dans une zone nommée zone de débordement. Le chaînage des enregistrements écrits en zone de débordement est alors mis à jour par le système.

Un des cas particuliers de l'organisation directe est l'organisation relative, pour laquelle la règle de calcul est la plus simple de toutes. L'adresse est en correspondance directe avec la valeur de clé. Ainsi, pour un fichier implanté à partir de l'adresse 1BC2 et dont l'enregistrement a une longueur de 30 octets :

30 (en base 10) = 1E (en hexadécimal)

clé = 0 le premier enregistrement est à l'adresse			1BC2
clé = 1 le second	1BC2 + 1E	=	1BE0
clé = 2 le troisième	1BE0 + 1E	=	1BFE

L'organisation relative est intéressante pour des fichiers qui
– possèdent une clé numérique;
– dont les valeurs de cette clé appartiennent à un petit intervalle;
– dont la grande majorité des entiers de cet intervalle existent comme valeurs de clé d'enregistrement.

REMARQUE

– Les supports qui permettent l'organisation relative sont les supports adressables.
– L'organisation relative permet d'utiliser les deux accès connus, séquentiel et direct.
– La clé est une variable entière non signée, elle n'appartient pas à la liste des rubriques de l'enregistrement.