Chapitre 2: Les fichiers séquentiels

Objectifs

- Comprendre les concepts de base relatifs aux fichiers
- Manipuler des fichiers à organisation séquentielle.

I. Notion de fichier

Dans tous les programmes que nous avons jusqu'à présent développés, lestockage des données se fait en mémoire vive qui est volatile et de capacité limitée. Or, la plupart des applications nécessitent une sauvegarde permanente des données. Pour éviter la perte de ces informations au débranchement de l'ordinateur, on utilise des ***fichiers.***

I.1. Définition

Un fichier est une structure de données formée de cellules contiguës permettant l'implantation d'une suite de données en mémoire secondaire (disque, disquette, CD-ROM, bande magnétique, etc.)
Chaque élément de la suite est appelé ***article*** et correspond généralement à un enregistrement.

Exemples :

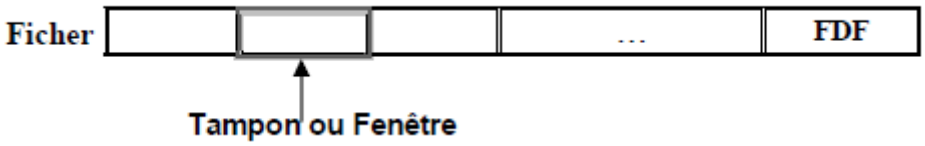
- liste des étudiants d'une institution
- état des produits stockés dans une magasin
- liste des employés d'une entreprise

I.2. Eléments attachés à un fichier

- On appelle ***nom interne*** d'un fichier le nom sous lequel un fichier est identifié dans un programme.
- On appelle ***nom externe*** d'un fichier le nom sous lequel le fichier est identifié en mémoire secondaire. Ce nom est composé de trois parties:
  - l'identifiant du support
  - le nom du fichier proprement dit
  - une extension (ou suffixe) qui précise le genre du fichier (donnée, texte, programme, etc.)

Ainsi, « **A:\nombres.DAT »** désigne un fichier de **données** (d'extension .dat) stocké sur le lecteur **A**:\ et qui s'appelle **nombres**.

- On appelle ***tampon*** ou ***buffer*** d'un fichier, une zone de la mémoire principale pouvant contenir un enregistrement du fichier. C'est une« fenêtre » à travers laquelle on « voit » le fichier.
- Un fichier possède toujours un enregistrement supplémentaire à la fin appelé ***marque de fin de fichier*** (FDF) permettant de le borner.



- Chaque fichier est caractérisé par :
  - un **mode d'organisation** : séquentielle, séquentielle indexée, relative ou sélective.
  - un **mode d'accès** : séquentiel ou direct

Un *fichier à organisation séquentielle* (F.O.S) ne permet que l'accès séquentiel : pour atteindre l'article de rang n, il est nécessaire de parcourir les (n-1) articles précédents.
L'accès direct se fait soit en utilisant le rang de l'enregistrement (cas de l'organisation relative) comme dans les tableaux, soit en utilisant une clé permettant d'identifier de façon unique chaque enregistrement (cas de l'organisation séquentielle indexée et sélective).

Remarque

Les caractéristiques d'un fichier sont étroitement liées aux langages de programmation qui offrent chacun différents types de fichiers.

II. Déclaration d'un fichier à organisation séquentielle

Pour déclarer une variable de type fichier, il faut spécifier :

- le nom du fichier
- le type des éléments du fichier

Exemple

```
Types
Structure Etudiant
        Numéro : Entier
        Nom : Chaîne[30]
        Prénom : Chaîne[30]
        Classe : Chaîne[5]
FinStruct

Fetud = Fichier de Etudiant

Variables
        Fe : Fetud
        et : Etudiant
```

III. Manipulation des fichiers à organisation séquentielle

Toute manipulation d'un fichier nécessite 3 phases :

1- Ouverture du fichier :

Ouvrir (NomFichier, mode)

Un fichier peut être ouvert en mode lecture (L) ou en mode écriture (E).
Après l'ouverture, le pointeur pointe sur le premier enregistrement du fichier.

2- Traitement du fichier :

Lire (NomFichier, fenêtre)

Cette primitive a pour effet de copier l'enregistrement actuel dans la fenêtre du fichier. Après chaque opération de lecture, le pointeur passe à l'enregistrement suivant. Si on veut lire une information en amont du pointeur, il faut rouvrir le fichier et le lire jusqu'à l'information désirée.

Ecrire (NomFichier, fenêtre)

Cette primitive a pour effet de copier le contenu de la fenêtre sur le fichier en mémoire secondaire. Dans un fichier à organisation séquentielle, l'ajout d'un nouvel article se fait toujours en fin de fichier.

3- Fermeture du fichier :

Fermer (NomFichier)

Remarque

La fonction booléenne **FDF(NomFichier)** permet de tester si la fin du fichier est atteinte. Sa valeur est déterminée par le dernier ordre de lecture exécuté.

III.1. Création d'un Fichier à organisation séquentielle

Ecrire une procédure permettant de créer et remplir un fichier d'étudiants.

```
Procédure Création (Var fe : Fetud)
Variables
        et : Etudiant
Début
        Ouvrir(fe,E)
        Ecrire ("donner le numéro de l'étudiant : ")
        Lire (et.Numéro)
        TantQue (et.Numéro # 0) Faire
            Ecrire ("Nom de l'étudiant : ")
            Lire (et.Nom)
            Ecrire ("Prénom de l'étudiant : ")
            Lire (et.Prénom)
```

```
        Ecrire ("Classe de l'étudiant : ")
        Lire (et.Classe)
        Ecrire (fe,et)
        Ecrire("donner le numéro de l'étudiant suivant: ")
        Lire(et.Numéro)
    FinTQ
    Fermer(fe)
FinProc
```

Dans cette procédure, l'utilisateur doit entrer la valeur 0 pour le champ numéro pour indiquer la fin de la saisie étant donné que le nombre d'étudiants n'est pas connu à l'avance.

III.2. Parcours d'un fichier à organisation séquentielle

Ecrire une procédure permettant d'afficher la liste des étudiants à partir d'un fichier d'étudiants.

```
Procédure Consultation (fe : Fetud)
Variables
    et : Etudiant
Début
    Ouvrir (fe,L)

    TantQue NON (FDF (fe) ) Faire
        Lire(fe,et)
        Ecrire(et.Numéro, et.Nom, et.Prénom, et.classe)
    FinTQ
    Fermer(fe)
FinProc
```

IV. Les fichiers de type texte

Les fichiers de texte sont des fichiers séquentiels qui contiennent des caractères organisés en lignes. La présentation sous forme de « ligne » est obtenue grâce à la présence des caractères de contrôle :

- retour chariot (noté souvent CR), de code ASCII 13
- saut de ligne (noté souvent LF) de code ASCII 10.

Un fichier texte peut être déclaré de 2 façons différentes :

Variables ftext : Fichier de Caractère	**Variables** ftext : Fichier Texte

Remarques

1- Un fichier de type texte peut être traité ligne par ligne ou caractère par caractère

2- Dans un fichier de type texte, la primitive **Lire_Ligne(NomFichier, Fenêtre)** permet de lire une ligne du fichier et la transférer dans la fenêtre.

3- De même, la fonction booléenne **FDL(NomFichier)** permet de vérifier si le pointeur a atteint la fin d'une ligne.

Exercice

Ecrire l'algorithme d'une procédure qui lit et affiche le contenu d'un fichier de type texte.

```
Procédure ParcoursFichText (ftext : Fichier texte)
Variables
    ligne : Chaîne
Début
    Ouvrir(ftext,L)
    TantQue Non(FDF(ftext)) Faire
        Lire_Ligne (ftext, ligne)
        Ecrire(ligne)

    FinTQ
    Fermer(ftext)
FinProc
```

V. Mise à jour d'un fichier séquentiel

Dans un fichier séquentiel, il est impossible de modifier directement un article, encore moins d'ajouter un nouvel article ailleurs qu'à la fin, et non plus de supprimer physiquement un article. Ces opérations de mise à jour ne peuvent se faire qu'en réécrivant complètement le fichier. La mise à jour directe n'est possible qu'avec les fichiers à accès direct.
Pour les fichiers de petite taille, voilà comment procéder :
- Copier tout son contenu en mémoire centrale, par exemple dans un tableau d'enregistrement. C'est le **chargement**.
- Faire les mises à jour désirées sur les données ainsi chargées en mémoire centrale
- Recopier ces données mises à jour dans le fichier initial (en écrasant ainsi les anciennes données). C'est la **sauvegarde**.

Pour mettre à jour des fichiers plus volumineux qui ne peuvent être intégralement chargés en mémoire centrale, on passe par l'intermédiaire d'un autre fichier.

VI. Exemple

```
Programme comptes_clients
Type
        Structure tcompte
            num: entier
            etat: caractère /*N pour normal, I pour impayé, C pour contentieux*/
            solde: réel
        finStruct
Var
        compte: tcompte
        cpteclt : fichier séquentiel de tcompte
        rep: caractère
        n, i : entier
        tabcompte : tableau [1..100] de tcompte
Début
/* création et remplissage du fichier */
        Ouvrir (cpteclt,E)
        // on saisit les informations sur les clients et on les écrit dans le fichier
        Répéter
                // on saisit les champs de l'enregistrement
                Ecrire (" donner le Numéro")
                Lire (  compte.num)
                Ecrire ("  donner l'etat")
                Lire ( compte.etat)
                Ecrire ("  donner  le solde")
                Lire (  compte.solde)
                // on recopie l'enregistrement dans le premier article vide du fichier
                Ecrire (cpteclt, compte)

                Ecrire (" Taper O si vous voulez ajouter un autre compte")
                Lire ( rep)

        Jusqu'à rep <> "O"
        Fermer (cpteclt)
```

```
        /* lecture des articles du fichier */
        Ouvrir (cpteclt, L)
        // On lit les articles et les affiche tant qu'on n'a pas atteint la fin du fichier
        Tantque NON(FDF(cpteclt)) Faire
        // récupérer l'article courant dans l'enregistrement en mémoire centrale
            Lire (cpteclt, compte)
           // afficher les champ de l'enregistrement
          Ecrire( "Numéro: ", compte.num)
          Ecrire ("Solde: ", compte.solde)
          Ecrire( "Etat:")
          Selon compte.etat Faire
             "N": Ecrire( "Normal")
             "I" : Ecrire ("Impayé")
             "C": Ecrire ("Contentieux")
         FinSelon
        FinTantque
        Fermer (cpteclt)
/* ajout d'un compte client à la fin du fichier */
        Ouvrir (cpteclt, E)
        // on saisit un enregistrement correspondant à l'article à ajouter
        Ecrire("Numéro?")
        Lire( compte.num)
        Ecrire(""etat?")
        Lire(  compte.etat)
        Ecrire(" "solde?")
        Lire( compte.solde)
        // on recopie l'enregistrement sur le fichier
        Ecrire (cpteclt, compte)
        Fermer (cpteclt)
/*modification d'un compte client, le 5689*/
//Chargement en mémoire centrale, dans le tableau d'enregistrements tabcompte et modification
        Ouvrir (cpteclt,L)
        n←  0
        Tantque NON (FDF(cpteclt)) Faire
            n←  n + 1
           Lire (cpteclt, tabcompte[n])
           Si tabcompte[n].num = 5689Alors
              tabcompte[n].solde ←  tabcompte[n].solde – 500
          FinSi
        FinTantque
// Sauvegarde du tableau dans le fichier
        Ouvrir (cpteclt, E)
        //n contient le nombre d'enregistrements
        Pour i de 1 à n Faire
        Ecrire (cpteclt, tabcompte[i])
        FinPour
        Fermer (cpteclt)
```

```
/*Suppression d'un compte, le 1268*/
//chargement
        Ouvrir (cpteclt, L)
        n←  0
        Tantque NON(FDF(cpteclt)) Faire
          n←  n + 1
          Lire (cpteclt, tabcompte[n])
        FinTantque
        Fermer (cpteclt)

//sauvegarde (sauf le compte 1268)
        Ouvrir (cpteclt, E)
        Pour i de 1 à n Faire
          Si tabcompte[i].num<> 1268 alors
              Ecrire (cpteclt, tabcompte[i])
          FinSi
         FinPour
        Fermer (cpteclt)

/* stockage dans un fichier séparé du numéro de tous les clients en contentieux.*/
// ajouter dans la partie déclaration des variables:
// cltcont : fichier séquentiel d'entiers // fichier contenant des numéros de clients
        Ouvrir (cpteclt, L)
        Ouvrir (cltcont, E)
        Tantque NON(FDF(cpteclt)) Faire
            Lire (cpteclt, compte)
            Si compte.etat = "C" Alors
               Ecrire (cltcont, compte.num)
            Finsi
        FinTantque
        Fermer(cpteclt)
        Fermer(cltcont)
Fin
```