Chapitre 07 : Les fichiers

I. Introduction

Imaginons que l'on veuille écrire un programme permettant d'enregistrer des renseignements (Nom, Prénom, adresse, téléphone) concernant nos clients. Ce programme doit également permettre de consulter et de modifier ces informations. Ces données ne peuvent donc pas être inclues dans l'algorithme, ni entrées au clavier à chaque nouvelle exécution.

Le problème qui se pose réside au niveau de la sauvegarde des informations après la fermeture du programme. Les variables qui sont des cases mémoires ne répondent pas à notre besoin à cause de leur disparition à chaque fin d'exécution.

Les fichiers sont là pour résoudre ce problème. Ils servent à stocker des informations sur un support de stockage (disquette, disque dur, CD Rom, etc.) de manière permanente pour les réutiliser ultérieurement.

II. Types de fichiers

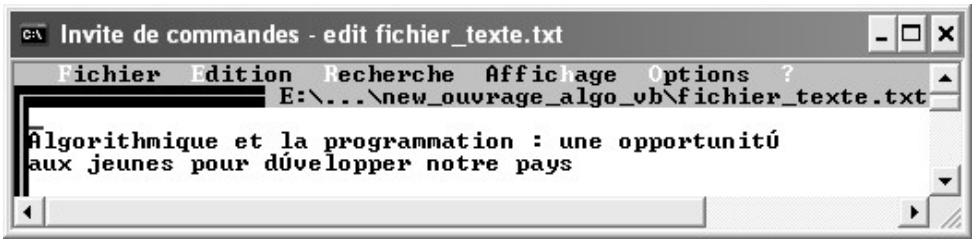
Le critère important qui différencie les fichiers est la façon dont les informations sont organisées sur ces derniers.

II.1. Les fichiers textes

Un fichier texte est formé de caractères ASCII, organisé en lignes, chacune se termine par un caractère de contrôle de fin de ligne.

Si chaque ligne contient le même genre d'informations, les lignes sont appelées des enregistrements. Par exemple, prenons le cas de départ, le fichier est destiné à stoker les coordonnées : nom, prénom, adresse et téléphone de chaque client. Dans ce cas, les informations concernant un client donné doivent être stockées sur une seule ligne.

Les fichiers texte peuvent être crées avec des éditeurs de texte et affichés de manière lisible à l'écran, voir la figure suivante :



II.2. Les fichiers binaires

Un fichier binaire contient des données non textuelles. Il n'est pas organisé sous forme d'enregistrement. Les fichiers binaires ne prennent sens que s'ils sont traités par un programme adapté. Par exemple un fichier son, une vidéo, une image, un programme exécutable, etc.

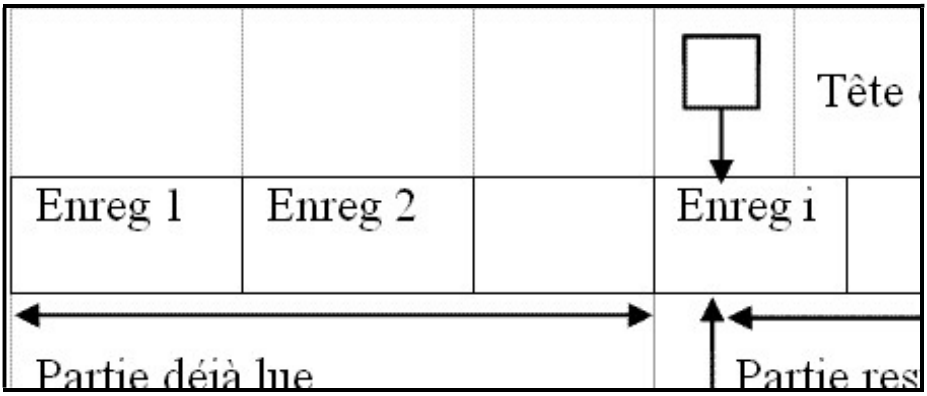
Dans les fichiers binaires, les données sont écrites à l'image exacte de leur codage en mémoire. Ceci facilite l'accès à ce type de fichier et le rend rapide, voir la figure suivante :

III. Types d'accès aux fichiers

Le type d'accès est la technique que la machine doit suivre pour aller chercher les informations contenues dans un fichier. On distingue trois types d'accès aux fichiers

➢ L'accès séquentiel :

Cet accès consiste à traiter les informations séquentiellement, c'est à dire dans l'ordre où elles apparaissent dans le fichier. On ne peut donc accéder à une information qu'en ayant au préalable examiné celle qui la précède. Le schéma suivant montre le principe de parcours d'un fichier à accès séquentiel.



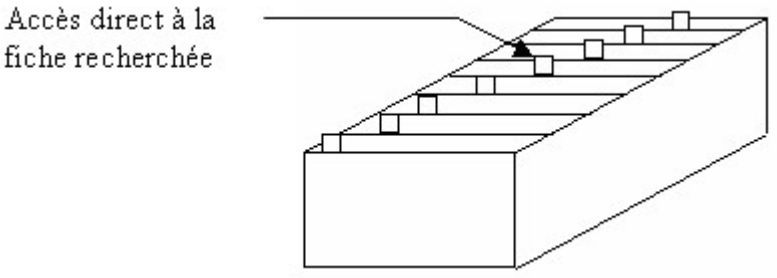
Remarques :

- La fin du fichier est repérée par un marqueur de fin du fichier.
- Pour ajouter une information, il faut que la tête de lecture /écriture se place en face du marqueur de fin de fichier.

➢ L'accès direct

Ce type d'accès consiste à se placer directement sur l'information souhaitée sans parcourir celles qui la précèdent, en précisant la position de l'élément recherché.

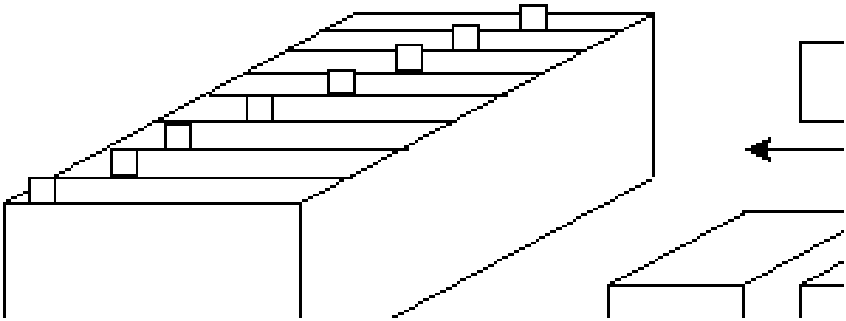
L'indication d'un numéro permet donc un accès direct et rapide à l'information ainsi référencée.



➢ L'accès indexé :

Ce type d'accès combine la rapidité de l'accès direct et la simplicité de l'accès séquentiel. Il est particulièrement adapté au traitement des gros fichiers, comme les bases de données.
Le principe est de créer des fichiers supplémentaires d'index (voir schéma ci-dessous).

Exemple : Fiche des livres



On parcourt un index pour rechercher une clef. On obtient ainsi l'adresse exacte de l'information recherchée. On peut utiliser des opérateurs de comparaisons sur les index (=, ≠, <, <=, >, >=). Il est alors possible, par exemple, de retrouver rapidement toutes les personnes de plus de 18 ans.
Dans l'exemple schématisé ci-dessus, on pourrait, grâce aux index, retrouver rapidement des livres à partir de leur auteur, de leur titre ou de leur thème.

IV. Traitement séquentiel des fichiers texte

IV.1. Ouvrir et fermer un fichier

➢ Ouvrir un fichier texte

Lorsqu'on désire accéder à un fichier, il est nécessaire avant tout accès, d'ouvrir le fichier.

Syntaxe :

Ouvrir Nom_du_Fichier en Num_canal en Mode

Nom_du_Fichier : c'est le nom physique du fichier.

Num_canal : c'est le nom logique du fichier. Pour ouvrir un fichier, il faut lui allouer un numéro du canal valide et disponible.

Mode : le mode d'ouverture du fichier conditionne le travail qui peut être effectué sur ses enregistrements. Il existe trois modes

d'ouverture du fichier texte :

- Lecture : permet d'ouvrir le fichier en lecture seul
- Ecriture : indique son accès en écriture. Dans ce mode, un nouveau fichier est toujours créé. Si le fichier existe déjà, il est réinitialisé à vide et son contenu précédent est perdu.
- Ajout : permet d'ajouter des données à un fichier séquentiel existant en conservant le contenu précédent.

Exemple :

```
// Num_canal égal à 1 si fiche.txt est le premier fichier à ouvrir.
// Il est égal à 2 si un fichier est déjà ouvert et fiche.txt est
// le deuxième fichier à ouvrir
Ouvrir "fiche.txt" en 1 en Lecture
```

Cette instruction ouvre le fichier ***fiche.txt*** en lecture seule. Le fichier à comme nom physique ***fiche.txt*** et comme nom logique un (1).

➢ Fermer un fichier texte

Une fois qu'on a terminé avec un fichier, il ne faut pas oublier de le fermer. On libère ainsi le canal qu'il occupait.

Syntaxe :

Fermer(Nom_du_Fichier)
Ou bien
Fermer (Num_canal)

Note :

Lorsqu'un fichier doit subir plusieurs interventions nécessitant plusieurs ouvertures, il sera nécessaire de fermer le fichier avant de le re-ouvrir.

IV.2. Lire et écrire dans un fichier

Soit un fichier texte contenant des données organisées dans des enregistrements de longueur fixe.

Champs	Nom	Prénom	Ville	Tél
Enreg 1	Mahboub	Khadija	Agadir	028435867
Enreg 2	Sabir	Ahmed	Marrakech	024251674
Enreg 3	Latif	Karima	Agadir	028457197

La lecture et l'écriture dans ce fichier exigent la déclaration de la structure suivante :

```
// Déclaration de la structure client
Type Structure client
   Nom : chaîne * 25
   Prénom : chaîne * 25
   Ville : chaîne * 15
   Tel : chaîne * 10
FinStruct
```

Après avoir défini la structure client, on peut l'utiliser pour créer une ou plusieurs variables correspondant à cette structure :

```
// Clt : variable de type client
Variables Clt : client
```

On peut maintenant remplir les différentes informations contenues au sein de la variable Clt de la manière suivante :

```
Clt.Nom ← "Mahboub"
Clt.Prénom ← "Khadija "
Clt.Ville ← "Agadir "
Clt.Tel ← "028435867"
```

Après avoir rempli les différents champs de la variable Clt, on peut utiliser cette variable pour écrire directement dans le fichier.

EcrireFichier 1, Clt

Pour une opération de lecture, il suffit de recopier un enregistrement dans une variable de type client et d'écrire la syntaxe suivante :

LireFichier 1, Clt

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

Exercice1 : Fichier Clients.txt

Ecrire un algorithme permettant de créer un fichier ***Clients.txt*** puis saisir les informations du 1er client et de les écrire dans ce fichier.

Solution :

```
Algorithme  Création_Fichier_Clients

// Déclaration de la structure client
Type Structure client
   Nom : chaîne * 25
   Prénom : chaîne * 25
   Ville : chaîne * 15
   Tel : chaîne * 9
FinStruct
// Clt : Variable de type client
Variable Clt : client

// Algorithme principal
Début
   // Création du fichier Clients.txt
   Ouvrir "Clients.txt" en 1 en Ecriture
   // Affectation de données aux champs de la structure
   Clt.Nom ← "Mahboub"
   Clt.Prénom ← "Khadija "
   Clt.Ville ← "Agadir "
   Clt.Tel ← "028435867"
   // Ecriture dans le fichier
   EcrireFichier 1, Clt
    // Fermeture du fichier
   Fermer (1)
Fin
```

Exercice 02 : Fichier clients.txt (suite)

En utilisant une procédure ***Affichage()***, écrire un algorithme permettant d'afficher à l'écran le contenu du fichier ***Clients.txt***. Cet algorithme complète l'exercice précédent.

Remarque :

La fonction booléenne ***EOF*** (acronyme pour End Of File) renvoie ***vrai*** si on a atteint la fin du fichier, sinon, elle renvoie ***faux***. Cette fonction est nommée parfois ***FinFichier***.

Solution :

```
Algorithme  Lecture_Fichier_Clients

// Déclaration de la structure client
Type Structure client
   Nom : chaîne * 25
   Prénom : chaîne * 25
   Ville : chaîne * 15
   Tel : chaîne * 10
FinStruct
Variable Clt : client

// Déclaration de la procédure Affichage
Procédure Affichage()
Début
   TanQue Non EOF(1)
      LireFichier, Clt
      Ecrire Clt
   FinTanQue
FinProc

// Algorithme principal
Début
   // Ouvrir le fichier Clients.txt
   Ouvrir "Clients.txt" en 1 en lecture
   // Appel de la procédure Affichage
   Affichage
   // Fermer le fichier
   Fermer (1)
Fin
```

On peut utiliser dans la procédure ***Affichage*** un tableau de structure dynamique dont le code est le suivant :

Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com

```
Variable Clts : Tableau[] de client

Procédure Affichage()
Début
   i ← -1
   Tantque Non EOF(1)
      i ← i + 1
      Redimensionner Clts[i]
      LireFichier 1, Clts[i]
      Ecrire Clts[i]
   FinTantQue
FinProc
```


Prof. Omar El kharki & Mme Jamila Mechbouh / elkharki@gmail.com