Chapitre 8 : *Les Fichiers*

Introduction

Toutes les données issues des programmes manipulés avant, étaient seulement stockées dans la RAM, ce qui fait que ces données sont volatiles. Pour sauvegarder ces données de façon permanente, dans des supports amovibles, les fichiers sont ainsi créés.

Les supports disposent d'organisation qui définissent la façon dont les données sont accessibles. Ces organisations peuvent être séquentielle, directe, relative, indexée etc.

Tout fichier à traiter doit être ouvert et fermé après traitement.

I. Les modes d'ouverture et primitives

Les modes d'ouverture permettent d'énoncer le type de traitement à appliquer à un fichier. Il existe trois modes d'ouverture :

A. ***Le mode d'ouverture en écriture** : Il permet d'ouvrir un fichier en mode création. S'il est appliqué à un fichier existant, alors son contenu sera écrasé. Sa syntaxe est :*

```
OUVRIR(nomVariablefichier) Ecriture
```

B. ***Le mode d'ouverture en Lecture** : Il permet d'ouvrir un fichier en mode consultation. Il doit être appliqué à un fichier existant. Sa syntaxe est :*

```
OUVRIR(nomVariablefichier)Lecture
```

C. ***Le mode d'ouverture en lecture/Ecriture** : Il permet d'ouvrir un fichier en mode Mise à jour c'est-à-dire il permet d'ajouter des données dans un fichier, d'y appliquer des modifications ou des suppressions. Sa syntaxe est :*

```
OUVRIR(nomVariablefichier) Lecture/Ecriture
```

Les modes d'ouverture \ Les primitives	***Ecriture***	***Lecture***	***Lecture/Ecriture***
Ecrire(tampon, nomVariablefichier)	*OUI*	*NON*	*OUI Ajout*
Lire(nomVariablefichier, tampon)	*NON*	*OUI*	*OUI Consultation*
Réécrire(tampon, nomVariablefichier)	*NON*	*NON*	*OUI Modification*
Supprimer(tampon, nomVariablefichier)	*NON*	*NON*	*OUI Suppression*

***Remarque** : Il existe des primitives associées à ces modes d'ouverture lesquelles permettant d'appliquer des traitements spécifiques à un fichier. Ces primitives sont résumées dans le tableau suivant :*

II. Déclaration de fichier

1. Syntaxe de déclaration de fichier de type primitif

Type nomFichier = Fichier typePrimitif

Organisation : séquentielle

var nomVariableFicher : nomFichier

***Exemple 1** : Fichier d'entiers à organisation séquentielle*

Type FEntiers = Fichier entier

Organisation : séquentielle

var Fe : FEntiers

***Exemple 2 :** Fichier de chaines à organisation séquentielle*

Type FChaines = Fichier chaine

Organisation : séquentielle

var F : FChaines

2. Syntaxe de déclaration de fichier d'enregistrements

Type nomEnreg = structure

DEBUT

Champ(s) : type(s)

FIN

Type nomFichier = Fichier nomEnreg

Organisation : sequentielle

var nomVariableFichier : nomFichier

***Exemple 1** : Fichier de personnes à organisation séquentielle. Personne (nom, prénom, sexe, âge)*

Type PERSONNE = structure

DEBUT

nom, prenom, : chaine

sexe : 'M', 'm', 'F', 'f'

age : entier

FIN

Type FPersonnes = Fichier PERSONNE

Organisation : sequentielle

var F : FPersonnes

III. Création de fichier

Pour créer un fichier il faut d'abord l'ouvrir en écriture ? récupérer les données à ajouter dans le fichier et les enregistrer avec la primitive Ecrire.

Exercice d'application 1 :

Ecrire un module qui permet de créer un fichier de personnes à organisation séquentielle. Les personnes sont saisies en fonction des besoins de l'utilisateur. Une personne est caractérisée par nom, prénom, sexe et âge.

Exercice d'application 2 :

Soit un tableau d'étudiants de T cellules. Ecrire un module qui créer 2 nouveaux fichiers l'un contenant tous les étudiants qui ont la moyenne et l'autre ce qui ne l'ont pas. Un étudiant = matricule, nom, prénom, classe, moyenne. T = 150.

IV. Consultation de fichier

Il est possible de visualiser le contenu d'un fichier afin d'y appliquer des traitements particuliers. Pour cela, le fichier doit être ouvert en mode lecture. Par défaut, le nombre d'enregistrements d'un fichier n'est pas connu à l'avance : il existe la fonction EOF(nomVariableFichier) ou FF(nomVariableFichier) qui renvoie VRAI si la fin du fichier est atteinte et FAUX dans le cas contraire.

***Application** :*

Soit un fichier de produits à organisation séquentielle. Ecrire un module qui affiche le contenu du fichier puis détermine le montant total en stock des produits. Un produit est caractérisé par son code, son nom, sa catégorie, son prix unitaire et sa quantité.

V. Les opérations de mise à jour

La mise à jour concerne trois opérations : l'ajout, la modification et la suppression.

1. L'opération d'ajout

L'ajout se fait toujours à la fin du fichier. Pour cela, il faut parcourir le fichier jusqu'à sa fin avant d'appliquer l'ajout.

APPLICATION

En considérant le fichier de produits défini dans l'application précédente, écrire un module qui reçoit un produit puis l'ajoute à la fin du fichier.

2. L'opération de modification

La modification d'un fichier s'applique à son contenu en fonction des critères posés. Pour appliquer la modification, il faudra utiliser la primitive REECRIRE(tampon, nomVariableFichier)

APPLICATION

En considérant le même fichier de produits, écrire un module qui réduit de 50% les prix unitaires des produits dont la catégorie contient ***éducation*** *et de 80% les prix unitaires des produits dont la catégorie contient* ***alimentaire****.*

3. L'opération de suppression

Elle permet de supprimer des données d'un fichier en fonction des paramètres recherchés

APPLICATION

Ecrire un module qui supprime du fichier tous les produits dont la quantité est égale à 0.