**Fiche synthèse sur les fichiers**

**Description**

Un fichier est un ensemble de données organisées en unités d’accès appelées « enregistrements » ou « articles » (tous du même type).

Un fichier est toujours enregistré sur un support externe à la mémoire centrale, c’est pour cela que les informations qu’il contient sont dites non volatiles. Les actions les plus courantes sur les fichiers sont :

* **Création :** définir le type du fichier ainsi que son emplacement physique.
* **Consultation :** exploiter les articles du fichier sans en modifier un seul.
* **Mise à jour :** ajouter, modifier ou supprimer des enregistrements ou des champs.

Le mode de représentation physique des fichiers varie d’un ordinateur à un autre. Ainsi, chaque langage de programmation gère les fichiers en fonction du système sur lequel il est installé.

**Le type Fichier**

Le langage pascal admet 3 classes : les fichiers **typés,** les fichiers **non typés**, les fichiers **texte**.

**Remarques :**

* Les fichiers typés et non typés sont des fichiers binaires dont les informations ne sont pas directement lisibles par l’être humain.
* La taille maximale d’un fichier est limitée par l’espace disponible sur son support, et sa manipulation dans un programme se fait à l’aide d’un pointeur.

**Déclaration des fichiers**

Soit « F » un fichier physique existant sur un support non volatile. Utiliser F dans un algorithme revient à lui associer une variable interne « f » par le biais de son chemin d’accès (de F). A savoir qu’un même fichier physique « F » peut voir ses données utilisées par plusieurs programmes.

**Les fichiers typés :**

Il y a deux façons de déclarer une variable interne « f » :

Syntaxe 1 : **Type** <identificateur\_fichier>=**fichier de** <type\_des\_articles>

**Var f** :<identificateur\_fichier>

Syntaxe 2 : **Var f** : **fichier de** <type\_des\_articles>

Ces deux déclarations permettent de définir des fichiers constitués du type qu’il désire, comme par exemple pour un fichier constitué uniquement de nombres entiers on écrira : **Var f : fichier de entier**. La taille maximale de « F » sera définie en fonction de l’espace disponible sur le disque.

**Note :** un fichier ne peut pas contenir d’autres fichiers.

**Les fichiers non typés :**

Très peu utilisés, ils permettent d’introduire dans un même fichier des articles appartenant à des types différents :

Syntaxe 1 : **Type** <identificateur\_fichier>=**fichier**

**Var f** :<identificateur\_fichier>

Syntaxe 2 : **Var f** : **fichier**

**Les fichiers Texte :**

Un fichier texte ne peut contenir que des informations de type texte. Constitué de plusieurs lignes, il se déclare à l’aide du mot-clé « Text » :

Syntaxe 1 : **Type** <identificateur\_fichier>=**texte**

**Var f** :<identificateur\_fichier>

Syntaxe 2 : **Var f** : **texte**

**Utilisation des fichiers**

**L’assignation :**

Comme dit précédemment, pour utiliser un fichier physique « F » dans un algorithme, il fallait que cet algorithme comporte une variable fichier « f ». Pour associer « f » et « F » on effectuera ce qu’on appelle une **assignation** pour que les modifications apportées à la variable « f » dans le programme affectent également « F » sur son support :

Syntaxe : **ASSIGNER(f,’chemin\_dacces\_F ‘)**

**Remarques :**

* **ASSIGNER()** est valable pour tous les types de fichiers.
* La même variable « f » peut effectuer des modifications sur deux fichiers physiques « F » et « G » à condition qu’ils soient de même type et qu’après avoir travaillé avec un fichier qu’il soit fermé avant d’assigner la variable au deuxième fichier.

**L’ouverture :**

* **OUVRIR(f)** si « F » est un fichier existant déjà sur le disque (ouvert en lecture).
* **REECRIRE(f)** si on veut créer un fichier « F » qui n’existe pas sur le disque, ou dans le cas où il existe, écraser tout son contenu (ouverture en écriture).
* **AJOUTER(f)** si on veut ouvrir un **fichier texte « F »** pour y ajouter des données en fin de fichier.

**Remarques :**

* **OUVRIR()** rend le fichier F consultable et le pointeur se positionne en début de fichier, mais permet uniquement la lecture du fichier.
* La variable booléenne **Eof(f)** permet de vérifier si le pointeur se positionne en fin de fichier ou non.

**La lecture et la mise à jour**

Après avoir ouvert un fichier, on peut lire les informations qu’il contient, en modifier certaines ou les supprimer, voire même en rajouter d’autres. Nous n’utiliserons ces possibilités que sur les fichiers typés et sur les fichier texte.

**Les fichiers typés :**

Un fichier typé « F » est constitué d’un ensemble d’articles du même type, sa lecture ou mise à jour dans l’algorithme nécessite une variable « p » ayant le type de ces articles.

Syntaxes :

* **Lire(f,p)** pour lire un article de « F » et insérer ses données dans la variable « p ».
* **Ecrire(f,p)** pour insérer dans « F » un article ayant les données contenues dans la variable « p », mais attention car si un article occupait déjà l’emplacement il y aura écrasement.

**Remarque :** après lecture ou écriture d’un article, le pointeur se positionne immédiatement sur l’article suivant. S’il n’y en a plus, le pointeur devient alors la fin de fichier.

**Les fichiers Texte :**

A l’exception de l’exemple d’avant, ce fichier étant exclusivement constitué de lignes de textes, la variable « p » sera obligatoirement une chaîne de caractères.

Syntaxes :

* **Lire(f,p)** pour lire une ligne de « F » et l’insérer dans la variable « p ».
* **Ecrire(f,p)** pour insérer **à la fin** de « F » une ligne de texte dont la valeur est contenue dans « p ».

**Remarques :**

* **Lire()** n’est possible que si le fichier texte a été ouvert en lecture avec OUVRIR(), ce qui positionne immédiatement le pointeur sur la ligne suivante après lecture ou devient la fin de fichier s’il n’y en a plus.
* **Ecrire()** n’est possible que si le fichier texte a été ouvert en écriture avec AJOUTER().
* Pour éviter les erreurs, il vaut mieux fermer le fichier avant de changer de mode d’ouverture.

**La fermeture**

Essentielle dans la manipulation des fichiers, elle permet d’éviter les erreurs d’entrée/sortie, de préserver l’intégrité des données d’un fichier ou encore d’optimiser un algorithme en priorisant les variables externes par rapport aux variables internes.

Syntaxe : **FERMER(f)**