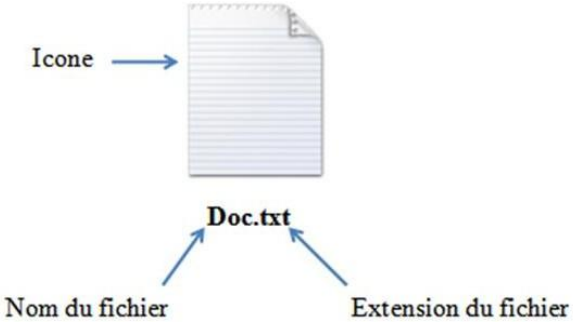
Chapitre 2 : les fichiers

1. Introduction

- Toutes les données utilisées jusqu'à maintenant étaient stockées en mémoire centrale, c'est-à- dire dans une zone volatile de la machine (RAM).
- La durée de vie des variables et des informations manipulées était donc égale au temps d'exécution du programme.
- Dans une entreprise, les informations doivent avoir une durée de vie supérieure à l'exécution d'un programme. Elles sont donc stockées sur des supports non volatiles (ex : disques dur, flache-disque, CD…). Les algorithmes peuvent grâce à des instructions précises manipuler les informations contenues dans ces fichiers.

2. Définition d'un fichier

- Il représente une ***unité*** d'***informations*** qui peut être stocké dans l'ordinateur, il peut être un programme, un texte, un son, une image, etc.
- Il est caractérisé par un ***nom***, et une ***extension***
- L'***extension*** permet d'identifier son ***type***.
- L'extension est composée d'un point suivi par un ensemble de caractères à la fin du nom de fichier.



- **Types de fichier :**

Le tableau suivant montre Certains types de fichiers et leurs extensions:

Types de fichier	Extension	Icone
Fichier exécutable	.exe	
Fichier image	.bmp, .gif, .jpg, .png	
Fichier texte	.txt (fichier de bloc note) .docx (fichier de Word)	
Fichiers Web	.html	
Fichier Programme	.cpp	
Fichier Base de données	.accdb	
Fichier Archive	.zip, .rar	

3. Fichier de données algorithmique :

En algorithmique, un fichier est un ensemble de données de même type, enregistrées sur un support de stockage permanent. Il est souvent d'utiliser le texte ou le type enregistrement (structure) pour stocker ces données.

3.1. Types de fichiers algorithmiques

En algorithmique, on a principalement deux types de fichiers :

1) **Le fichier texte :** Ils sont constitués d'une suite de caractères formant un texte (chaîne de caractères). On les utilise pour enregistrer des textes mais également des valeurs numériques en vue de les échanger avec d'autres logiciels. Ils sont lisibles par un simple éditeur de texte.
2) **Le fichier binaire :** contenant des données sous forme d'octets qui n'ont donc de sens que pour le logiciel qui les utilise, ce type de fichiers est illisible par un éditeur de texte, il est constitué d'une collection d'enregistrements, chaque enregistrement contenant une collection d'unités logiques d'informations encore appelées champs.

3.2. Manipulation des fichiers

La plupart des langages de programmation actuels disposent d'instructions permettant la manipulation de fichiers. Ces instructions peuvent être classées comme suit:

- Ouverture et création d'un fichier,
- Fermeture d'un fichier
- Lecture et écriture d'enregistrements du fichier,
- Positionnement dans le fichier,
- Détection de la fin du fichier.

Remarque importante :

Pour utiliser un fichier physique **F** dans un algorithme, il fallait que cet algorithme comporte une variable fichier **f**. L'association entre **f** et **F** s'effectuera donc au moyen d'un procédé appelé **assignation**, de telle sorte que les modifications apportées à **f** dans l'algorithme affecteront directement **F** sur son support.

Syntaxe : **ASSIGNER (f, ' chemin_dacces_F ') ;**

a) Ouverture et création d'un fichier

Après l'assignation, il faut toujours ouvrir un fichier pour pouvoir l'utiliser, L'instruction d'ouverture du fichier doit indiquer si le fichier va être lu, modifié ou créé.

Syntaxes :

- **OUVRIR(f) ;** // si **F** est un fichier existant déjà sur disque (ouverture en lecture).
- **REECRIRE(f) ;** // si on veut créer un fichier **F** qui n'existe pas encore sur disque, ou bien qui existe déjà mais dont on veut écraser tout le contenu (ouverture en écriture).
- **AJOUTER(f) ;** // si on veut ouvrir un fichier **F** en pouvant y ajouter des données(uniquement en fin de fichier).

Remarque :

- **Ouvrir ( )** rend **F** consultable et positionne le pointeur en début de fichier.

- Dans le cas où **F** n'existe pas sur le disque, **REECRIRE ( )** permet de créer un fichier dont le nom et le chemin d'accès sont ceux précisés pendant l'assignation.
- Il peut arriver qu'on ouvre un fichier qui ne contient aucune donnée. Le début de fichier sera alors la fin de fichier. La fonction booléenne **Eof(f)** ; **E**nd **o**f **f**ile; permet à tout moment de vérifier si le pointeur est arrivé à la fin du fichier ou non.
-

b) Lecture et écriture d'enregistrements du fichier

Après l'ouverture du fichier, plusieurs possibilités sont offertes : lire les informations qu'il contient, en modifier quelques-unes, en supprimer certaines, ou alors en ajouter d'autres.

Syntaxes :

Lire (f, p) ; pour lire un enregistrement de **F** et insérer les données qu'il contient dans la variable **p** (p est de type enregistrement).

Ecrire (f, p) ; pour insérer dans **F** un enregistrement ayant les données contenues dans la variable **p** (p est de type enregistrement).

Remarque :

Après la lecture/écriture d'un enregistrement, le pointeur se positionne immédiatement sur l'enregistrement suivant. S'il n'y en a plus, la position du pointeur devient alors la fin de fichier.

c) Fermeture d'un fichier

La fermeture est une opération essentielle dans la manipulation des fichiers. Un bon programmeur ne doit donc jamais la négliger, car elle permet d'éviter les erreurs d'entrée/sortie, de préserver l'intégrité des données d'un fichier, d'optimiser un algorithme en utilisant un minimum de variables internes avec un maximum de fichiers externes.

Syntaxe : **FERMER**(f) ;

Exemple :

Écrire un algorithme qui permet d'afficher toutes les valeurs contenues dans un fichierd'entiers F.

Algorithme fichiersp : entier ;

f : fichier de entier ;

Début

Assigner (**f**, '**c:\entiers.txt**') ;Ouvrir(**f**) ;

tant que (NON (**Eof(f)**)) faireLire (f, p) ;

Ecrire ('Nous

venons de lire le

nombre ', p) ;

FINTANTQUE

FERMER(f) ;

Fin.