Les enregistrements et les fichiers

Pr. Badraddine AGHOUTANE
b.aghoutane@umi.ac.ma

Introduction :

Les **langages de programmation** offrent d'autres types de **structures de données** appelés **enregistrements**.

- Un **enregistrement** est composé de plusieurs champs (donnée élémentaire).
- C'est un **regroupement de données** qui doit être considéré ensemble.

Exemples d'enregistrements : les fiches des étudiants.

- Chaque **fiche d'étudiant** est caractérisée par les champs suivants : **nom**, **prénom**, **numéro d'inscription**.
- Donner trois autres exemples d'objets du monde réel ainsi que certaines de leurs caractéristiques (champs) :

Objet	Caractéristiques (propriétés)
Etudiant	Nom, prénom, numéro d'inscription
Patient	Code, Prénom, Nom, Taille, Poids, Genre
Voiture	Matricule, Puissance, Couleur, marque, Année de circulation
Produit	référence, nom, prix

Déclaration des enregistrements

En algorithmique :

```
Enregistrement FicheEtudiant
        nom, prenom : chaine de caractères
        numero : entier
FinEnregistrement
```

- Un enregistrement est un type comme les autres types.
- Ainsi la déclaration suivante :

f, g : FicheEtudiant

Définit **deux enregistrements (variables)** « f » et « g » de type **FicheEtudiant**

- L'enregistrement **FicheEtudiant** contient plusieurs champs, on y accède par leur nom précédé d'un point « . » :
  - **f.nom** désigne le champ (de type chaine) nom de la fiche f.
  - **f.numero** désigne le champ (de type entier) numéro de la fiche f.

Définition des enregistrements

- Pour définir les champs d'un enregistrement, on écrit :

```
f : FicheEtudiant
f.nom ← "XXXXX"
f.prenom ← "YYYYY"
f.numero ← 1256
```

- Les affectations entre enregistrement se font champ par champ.

Exemple d'enregistrements

- Un **catalogue de produits** dans un magasin. Un **article** est décrit par une **référence**, un **nom** et un **prix**.

```
Enregistrement article
        ref : chaîne de caractère
        nom : chaîne de caractère
        prix : Réel
FinEnregistrement
```

- L'utilisation des enregistrements dans un algorithme se fait comme suit :

```
Algorithme GestionArticles
Variables
    a1 : article
Début
        Ecrire(" Entrez la référence du premier article ? ")
        Lire (a1.ref)
        Ecrire(" Entrez le nom du premier article ? ")
        Lire (a1.nom)
        Ecrire(" Entrez le Prix du premier article ? ")
        Lire (a1.prix)
        Ecrire(a1.ref, a1.nom, a1.prix)
Fin
```

Exercice d'application :

Soit l'enregistrement **Patient** suivant formé de 5 champs, on vous demande de :

1- Déclarer en algorithmique l'enregistrement **Patient**

patient

Champ	Libélle	Type
C	Code	Entier
P	Prénom	Chaine
N	Nom	Chaine
T	Taille en mètres	Réel
M	Poids en Kilogrammes	Réel
G	Genre	Caractère

2- Déclarer en algorithmique deux variables **P1** et **P2** de type **patient**.

3- Remplir les champs des patients P1 et P2 par des valeurs de votre choix

4- Afficher pour chaque patient son Code, Nom, Prénom et son IMC (Indice de masse corporelle) sachant que :

$IMC = M/T^2$ (P= poids en Kilogrammes et T=taille en metres)

5- Traduire les questions 1,2,3 et 4 en python

Les fichiers : introduction

Remarques

- Lorsque l'on utilise un programme :
  - on saisit des informations,
  - on ≪ calcule ≫ des nouvelle informations
- À l'arrêt du programme ces informations sont perdues

- L'objectif des fichiers est de conserver ces informations d'une exécution à l'autre du programme
- Cela amène à **enregistrer** de l'information et à **relire** de l'information

Les fichiers

- Un **fichier** (*anglais : file)* est un **ensemble structuré de données de même type**.
- Il sert à **stocker** des informations de **manière permanente** afin de permettre une réutilisation ultérieure des informations qu'il contient
- À l'encontre des **variables** qui sont stockés dans des **mémoires vives**, les **fichiers**, eux sont stockés sur des **mémoires non volatiles** (*disque dur, flash disque, DVD, …*).
- Un **fichier** peut contenir des caractères (*fichier textes*), des programmes, des valeurs (*fichier de données*)
- Un fichier se distingue des autres composants par quelques attributs: son **nom** et sa **catégorie (binaire, texte)**.
- Les fichiers possèdent un **format** qui indique comment sont organisées les informations. L'extension suffixe du nom du fichier indique le format (par exemple .txt, .pdf, .docx)

Les catégories des fichiers

Il existe **deux catégories** de fichiers :

1. **Fichiers textes :**
   - Les fichiers textes sont les fichiers normaux (*toutes les données sont écrites sous forme de texte*).
   - On peut facilement créer des fichiers texte à l'aide des éditeurs de texte (*Bloc notes, MS word,…*).
   - Ils contient le même genre d'information sur chaque ligne. Ces lignes sont appelées des **enregistrements**.
   - Lorsqu'on ouvre ces fichiers, on peut voir tout le contenu du fichier sous forme de texte brut.
   - On peut facilement modifier ou supprimer le contenu.
2. **Fichiers binaires :**
   - Les fichiers binaires stockent les données sous forme binaire (*0 et 1*).
   - Ils peuvent contenir une grande quantité de données,
   - ils ne sont pas lisibles facilement et offrent une meilleure sécurité que les fichiers textes.
   - Ce sont des fichiers qui ne possèdent pas de structure de lignes (d'enregistrement).

   Exemple : fichier son, image, vidéo, programme exécutable, etc.

Organisation : Fichiers texte (exemple)

Un fichier peut être structuré en enregistrement (par exemple : Fiches des étudiants).

- Ce fichier est destiné à mémoriser les **coordonnées d'un certain nombre d'étudiants**.
- Pour **chaque étudiant**, il faudra noter le **nom**, **prénom**, **numéro** et **l'email**.
- Il paraît plus simple de stocker un étudiant par ligne du fichier (par enregistrement).
- Autrement dit, une ligne ne contient que les informations concernant un étudiant.
- Un fichier ainsi codé sous forme d'enregistrements est appelé un **fichier texte**.
- Entre chaque enregistrement, sont stockés les octets relatifs aux caractères signifiant un retour au début de la ligne suivante.

Structures des enregistrements fichiers

Les **fichiers** peuvent être structurés en **enregistrements**.

Il y a deux grandes possibilités pour structurer ces enregistrements :

1. **La structure n°1 est dite délimitée :** Elle utilise un **caractère spécial**, appelé caractère de séparation ou délimitation, qui permet de repérer la fin d'un champ et le début d'un autre.

Exemple :

- "Hassnaoui"; "Mouad"; 0123; "h.mouad@yahoo.fr"
- "Idrissi"; "Jalal"; 0456; "i.jalal@gmail.com"

- **La structure n°2 est dite à champs de largeur fixe :** il n'y a pas de délimiteurs. Chaque **champ a une longueur prédéfinie** et occupe toute cette longueur, sinon il est complété par des espaces.

Exemple :

• Hassnaoui	Mouad	0123	h.mouad@yahoo.fr
• Idrissi	Jalal	0456	i.jalal@gmail.com

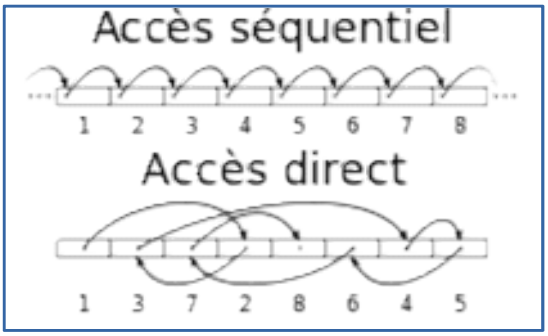
Types d'accès aux fichiers

Le **type d'accès** est la manière avec laquelle la machine **cherchera les informations** incluses dans le fichier. Le curseur indique la position de la prochaine lecture ou écriture

- **L'accès séquentiel** : Commence au début du fichier. Pour lire une information particulière, il faut lire toutes les informations précédentes. Dans le cas d'un fichier texte, cela signifie qu'on lit le fichier ligne par ligne (*enregistrement par enregistrement*).

- **L'accès direct** : On peut accéder directement à l'information désirée (*enregistrement*), en précisant le numéro d'emplacement (*numéro d'ordre*) de cette information.

  ➔ **L'accès direct** est évidemment préférable à l'accès séquentiel lorsqu'une application doit écrire, lire ou modifier un seul élément d'un large ensemble.



Les modes d'ouvertures des fichiers

- Si on veut travailler sur un fichier, la première chose à faire après la **déclaration** est de l'**ouvrir**.
- Lorsqu'on ouvre un fichier, il faut **exprime** ce qu'on va en faire: **lire, écrire** ou **ajouter.**

Il existe **quatre mode d'ouverture** d'un fichier :

1. **Lecture (position du curseur en début de fichier) :**
   - On peut uniquement accéder et récupérer les informations qu'il contient,
   - On ne peut pas modifier son contenu.
2. **Ecriture (position du curseur en début de fichier) :**
   - On peut mettre dedans toutes les informations que l'on veut.
   - Mais les informations précédentes, si elles existent, seront **intégralement écrasées** .
   - On ne pourra pas accéder aux informations qui existaient précédemment.
3. **Lecture/écriture (position du curseur en début de fichier) :**
   - Les données sont préservées, et il est possible de modifier le contenu du fichier.
4. **Ajout (position du curseur en fin de fichier) :**
   - On ne peut ni lire, ni modifier les informations existantes.
   - Mais, on peut ajouter de nouvelles lignes (*nouveaux enregistrement)* à la fin du fichier.

- Les **exemples** suivants concernent le type de base : **fichier texte en accès séquentiel**

Déclaration et ouvertures des fichiers séquentiels

La **structure fichier** se déclare comme une **variable** avec un type prédéfini :

- **Variable nom_fichier : Fichier**

Pour **ouvrir un fichier texte**, on écrira par exemple :

- **nom_fichier ← "Toto.txt"**
- **Ouvrir (nom_fichier) en Lecture**

  ou bien
- **Ouvrir (nom_fichier, mode d'ouverture)**

Lecture, écriture et fermeture des fichiers

Lire\ écrire dans un fichier ouvert :

- **Lire_Fichier (nom_fichier,** liste de variables**)**
- **Ecrire_Fichier (nom_fichier,** liste de variables**)**

Fermer le fichier :

- **Fermer (nom_fichier)**

Tester la fin du fichier :

- **EOF (nom_fichier)** *# EOF: End Of File*

Exemple d'ajout dans un fichier

```
Algorithme ExempleFichier
Variable
        Truc : chaîne Caractère
        Fich : Fichier
Début
        Fich ← "Exemple.txt"
        Truc ← "Fonfec"; "Sophie"; 0142156487; fonfec@yahoo.fr;
        Ouvrir(Fich) en Ajout
        Ecrire_Fichier (Fich, Truc)
        Fermer(Fich)
Fin
```

Exemple de lecture d'un fichier

```
Variable
        Truc : chaîne Caractère
        Fich : Fichier
Début
        Fich ← "Exemple.txt"
        Ouvrir(Fich) en Lecture
        Tantque Non EOF(Fich) Faire
                Lire_Fichier (Fich,Truc)
        FinTantQue
        Fermer(Fich)
Fin
```

Algorithmique – fichiers séquentiels

Exercice :

On souhaite mémoriser des noms des personnes dans un fichier nommé « **personne.txt** », créer les sous-programmes qui suivent :

1. Une procédure de création du fichier qui contient les noms des personnes.
2. Une procédure d'affichage des noms de personnes.
3. Une fonction qui permet de chercher un nom passé en argument et qui renvoie vrai si ce dernier est existant et faux sinon.

Ecrire un algorithme principal faisant appel aux différents sous-programmes.

1. Une procédure de création du fichier qui contient les noms des personnes.

```
Procedure Creation(fn : Fichier)
        Variable    n : chaine,   rep : caractère
Début
        fn ← Ouvrir (personne.txt, ECRITURE)
        Rep← 'O'
        Tant que (rep = 'O') Faire
                Ecrire("Entrer un Nom : "),
                Lire(n)
                Ecrire_Fichier(fn,n)
                Ecrire("Voulez-vous ajouter un autre nom (O/N) : ")
                Lire(rep)
        Fin Tant que
        Fermer(fn)
FinProcedure
```

2. Une procédure d'affichage des noms de personnes.

```
Procedure  Affichage(fn : Fichier)
        Variable n : chaîne
Debut
        fn← Ouvrir(personne.txt, LECTURE)
        Tant que NON(EOF(fn)) Faire
                Lire_Fichier(fn,n)
                Ecrire(n)
        Fin Tant que
        Fermer(fn)
FinProcedure
```

3. Une fonction qui permet de chercher un nom passé en argument et qui renvoie vrai si ce dernier est existant et faux sinon.

```
Fonction  Recherche(x : chaine ; fn : Fichier) : Booleen
        Variable n : chaine, Trouve : Booleen
Debut
        fn← Ouvrir(personne.txt, LECTURE)
        Trouve=faux
        Tant que ((Trouve=faux) ET (NON(EOF(fn))) Faire
                Lire_Fichier (fn,n)
                Trouve ← n = x
        Fin Tant que
        Si (EOF(fn)) Alors
                Retourne Faux
        Sinon
                Retourne Vrai
        Fin Si
        Fermer(fn)
FinFonction
```

4. Ecrire un algorithme principal faisant appel aux différents sous-programmes.

```
Algorithme personne
        Variable F1:Fichier
Debut
        Creation(F1)
        Affichage(F1)
        Si Recherche("Riadh", F1) Alors
                Ecrire("Riadh est existant dans le fichier")
        Sinon
                Ecrire("Riadh est non existant dans le fichier")
        Fin Si
Fin
```