Informatique

Chloé Cabot

novembre 2020

CPII ESIGELEC 2A

Rappels

Les enregistrements

- Enregistrement = structure de données regroupant un ensemble de champs, de **types différents**

```
Ennemi : ENREGISTREMENT
    nom : tableau de 30 caractères
    pt_vie : entier
    arme : tableau de 30 caractères
FIN ENREGISTREMENT
```

- On accède à un champ avec le « . » (point)

```
gargantua : Ennemi
gargantua.pt_vie = 100
```

- On peut imbriquer des enregistrements

```
Map : ENREGISTREMENT
    dimension : entier
    ennemis : tableau de Ennemi
FIN ENREGISTREMENT
```

Les fichiers

Introduction

Comment sauvegarder sur l'ordinateur les données saisies afin de les conserver une fois le programme terminé ? Par exemple quand on saisit une collection de vinyles ou des notes ?

Dans des fichiers

Introduction

- Les fichiers permettent la **persistence** des données
- On peut stocker du texte ou des nombres
- Et des enregistrements ! Les fichiers sont alors **structurés**

Les fichiers

La manipulation d'un fichier se fait par l'intermédiaire d'une variable. Exemple :

```
# Déclaration d'un enreg. Personne :
Personne : ENREGISTREMENT
    nom : tableau de 30 caractères
    age : entier
FIN ENREGISTREMENT
# Déclaration d'une variable de type fichier de Personne :
fic : fichier de Personne
```

Opérations sur les fichiers

Plusieurs opérations sont réalisables sur les fichiers :

- l'ouverture
- la fermeture
- la lecture
- l'écriture

Opérations sur les fichiers : ouverture

Avant d'effectuer des opérations sur un fichier, il est nécessaire de l'ouvrir. L'ouverture dépend de ce que l'on veut faire avec le fichier :

- créer le fichier sur l'ordinateur pour écrire dedans,
- ajouter des enregistrements dans le fichier,
- lire le contenu du fichier.

Opérations sur les fichiers : ouverture en création

Crée un fichier vide, et permet d'écrire dedans. Si le fichier existait déjà, le contenu dans l'ancien est supprimé.

```
ALGORITHME ouvrir_en_création
  entrées
     nom_fichier : tableau de caractères
  SORTIE
      fichier d'enregistrements
```

Exemple :

```
  fic ← ouvrir_en_création("personnes.dat")
```

Opérations sur les fichiers : ouverture en ajout

Le fichier est ouvert et de nouveaux enregistrements peuvent être ajoutés à la fin.

```
ALGORITHME ouvrir_en_ajout
  entrées
     nom_fichier : tableau de caractères
  SORTIE
      fichier d'enregistrements
```

Exemple :

```
  fic ← ouvrir_en_ajout("personnes.dat")
```

Opérations sur les fichiers : ouverture en lecture

Le fichier est ouvert et il est possible de lire son contenu à partir du début.

```
ALGORITHME ouvrir_en_lecture
  entrées
     nom_fichier : tableau de caractères
  SORTIE
      fichier d'enregistrements
```

Exemple :

```
  fic ← ouvrir_en_lecture("personnes.dat")
```

Opérations sur les fichiers : lecture

Permet de lire un enregistrement dans le fichier, et déplace le curseur de lecture.

```
ALGORITHME lire
  entrées
     fic : fichier d'enregistrements
  SORTIE
      enregistrement
```

Exemple :

```
  perso : Personne
  perso ← lire(fic)
```

Opérations sur les fichiers : écriture

Permet d'écrire un enregistrement dans le fichier, et déplace le curseur de lecture.

```
ALGORITHME écrire
  entrées
     fic : fichier d'enregistrements
     enreg : enregistrement
  SORTIE
      néant
```

Exemple :

```
  perso : Personne
  écrire(fic, perso)
```

Opérations sur les fichiers : fermeture

Le fichier doit être fermé à la fin du programme. S'il a été ouvert en lecture ou ajout, il est nécessaire de le fermer si on a besoin de lire dedans (et réciproquement).

```
ALGORITHME fermer
  entrées
     fic : fichier d'enregistrements
  SORTIE
      néant
```

Exemple :

```
  fermer(fic)
```

Fin de fichier

Lors de la lecture il est généralement nécessaire de savoir si on est arrivé à la fin. La fonction **`eof`** (End Of File) indique si on est ou non à la fin du fichier :

```
ALGORITHME eof
  entrées
     fic : fichier d'enregistrements
  SORTIE
     booléen
```

Exemple :

```
  fini : booléen
  fini ← eof(fic) # Vrai si on est à la fin, faux sinon
```

Remarques

- Il existe une fonction permettant de se déplacer dans un fichier, mais nous déconseillons de l'utiliser.
- Pour les opérations qui nécessitent de combiner lecture et écriture (modification, suppression d'un enregistrement, etc...) il est nécessaire d'utiliser un tableau.

Synthèse

- Les fichiers permettent de sauvegarder des données après la fermeture du programme (persistence)
- On manipule un fichier par l'intermédiaire d'une variable
- 4 opérations réalisables sur des fichiers :
  - Ouverture : en création, en ajout ou en lecture
  - Lecture
  - Écriture
  - Fermeture
- La fonction `eof` indique si on est arrivé en fin de fichier

Exercice

On se propose de réaliser un algorithme permettant de gérer la paye des employés d'une entreprise. Un employé est caractérisé par :

- son matricule,
- son nom,
- son prénom,
- son total d'heures travaillées dans le mois,
- son salaire horaire.

Écrire l'enregistrement permettant de représenter un employé

Enregistrement Employe

```
Employe : ENREGISTREMENT
   matricule : entier
   nom : chaine de 30 caractères
   prenom : chaine de 30 caractères
   total_heures : entier #on ne compte pas les demi-heures
   taux_horaire : réel
FIN ENREGISTREMENT
```

Exercice

On travaillera sur le fichier ”employe.dat”. C’est l’algorithme principal qui, à chaque lancement, vérifiera si le fichier existe ou non. Dans le cas où le fichier n’existe pas, c’est l’algorithme principal qui se chargera de le créer.

```
Employe : ENREGISTREMENT
   matricule : entier
   nom : chaine de 30 caractères
   prenom : chaine de 30 caractères
   total_heures : entier #on ne compte pas les demi-heures
   taux_horaire : réel
FIN ENREGISTREMENT
```

Écrire l’algorithme de la fonction permettant d’ajouter un employé

Ajout d'un employé

```
ALGORITHME ajout
   ENVIRONNEMENT
      Entree :
         néant
      Sortie :
         néant
      Interne :
         e : Employe
         ref_fichier : FICHIER DE Employe
```

Ajout d'un employé

```
   TRAITEMENT
   DEBUT
      #on initialise e champ par champ
      ECRIRE "matricule de l'employe?"
      e.matricule ← lire
      ECRIRE "nom de l'employe?"
      e.nom ← saisir(e.nom)
      ECRIRE "prénom de l'employe?"
      e.prenom ← saisir(e.prenom)
      #puisque l'employé arrive dans la base
      e.total_heures ← 0
      ECRIRE "taux horaire de l'employé?"
      e.taux_horaire ← LIRE
      #on peut désormais l'enregistrer
      ref_fichier ← OUVRIR_EN_AJOUT("employe.dat")
      ECRIRE(ref_fichier, e)
      FERMER(ref_fichier)
   FIN
FIN ALGORITHME
```

Exercice

Écrire l'algorithme de la fonction permettant d'afficher les caractéristiques de tous les employés

- Commencez par écrire une fonction qui affiche les caractéristiques d'un seul employé
- Écrivez une seconde fonction qui affichera les caractéristiques de tous les employés du fichier

Affichage d'UN employé

```
ALGORITHME aff_employe
   ENVIRONNEMENT
      entree:
         e : Employe
      sortie:
         néant
      interne:
         néant
   TRAITEMENT
      DEBUT
         ECRIRE "matricule : "
         ECRIRE e.matricule
         ECRIRE "nom : "
         afficher(e.nom)
         ECRIRE "prénom : "
         afficher(e.prenom)
         ECRIRE "total heures travaillées : " , e.total_heure
         ECRIRE "taux horaire de l'employé : " e.taux_horaire
      FIN
FIN ALGORITHME
```

Affichage de tous les employés

```
ALGORITHME aff_all_employes
   ENVIRONNEMENT
      entree :
         néant
      sortie :
         néant
      interne :
         ref_fichier : FICHIER DE Employe
   TRAITEMENT
      DEBUT
         #On commence par ouvrir le fichier
         ref_fichier ← OUVRIR_EN_LECTURE("employe.dat")
         TANT QUE (NON EOF(ref_fichier)) FAIRE
            #on appelle la fonction qui affiche 1 Employe pour
    chaque Employe du fichier
            aff_employe(LIRE(ref_fichier))
         FIN TANT QUE
         FERMER(ref_fichier)
      FIN
FIN ALGORITHME
```

Exercice

Écrire l'algorithme de la fonction permettant de supprimer un employé du fichier

Suppression d’un employé

```
ALGORITHME suppr_employe
   ENVIRONNEMENT
      entree :
         matricule_recherche : entier
      sortie :
         néant
      interne :
         ref_fichier : FICHIER DE Employe
         tab : tableau de 1000 Employe #on réserve l'espace
    temporaire en RAM
         e : Employe
         compteur : entier
         index : entier
```

```
   TRAITEMENT
      DEBUT
          compteur ← 0
          #On commence par ouvrir le fichier
          ref_fichier ← OUVRIR_EN_LECTURE("employe.dat")
          TANT QUE (NON EOF(ref_fichier)) FAIRE
             e ← LIRE(ref_fichier)
             SI (e.matricule != matricule_recherche) FAIRE #on ne
     copiera pas l'employé recherché
                tab(compteur) ← e
                compteur ← compteur + 1 #on veut écrire dans la
     case suivante au prochain Employe
             FIN SI
          FIN TANT QUE
          FERMER(ref_fichier)
          #on peut maintenant écraser le fichier original pour y
     écrire le contenu du tableau
          ref_fichier ← OUVRIR_EN_CREATION("employe.dat")
          POUR index ALLANT DE 0 A (compteur - 1) PAR PAS DE 1
     FAIRE
             ECRIRE(ref_fichier, tab(index))
          FIN POUR
          FERMER(ref_fichier)
      FIN
FIN ALGORITHME
```

Exercice

Écrire l'algorithme de la fonction permettant de mettre à jour le nombre d'heures travaillées depuis le début du mois

Mise à jour des heures

```
ALGORITHME maj_h_employe
   ENVIRONNEMENT
      entree :
         matricule_recherche : entier
      sortie :
         néant
      interne :
         ref_fichier : FICHIER DE Employe
         tab : tableau de 1000 Employe #on réserve l'espace
    temporaire en RAM
         e : Employe
         compteur : entier
         index : entier
```

```
   TRAITEMENT
      DEBUT
      compteur ← 0
      #On commence par ouvrir le fichier
      ref_fichier ← OUVRIR_EN_LECTURE("employe.dat")
      TANT QUE (NON EOF(ref_fichier)) FAIRE
         e ← LIRE(ref_fichier)
         SI (e.matricule = matricule_recherche) FAIRE
            ECRIRE "nombre d'heures actuel : " , e. total_heures
            ECRIRE "nouveau nombre d'heures ?"
            e. total_heures ← LIRE
         FIN SI
         tab(compteur) ← e
         compteur ← compteur + 1
      FIN TANT QUE
      FERMER(ref_fichier)
      #on peut maintenant écraser le fichier original pour y écrire
    le contenu du tableau
      ref_fichier ← OUVRIR_EN_CREATION("employe.dat")
      POUR index ALLANT DE 0 A (compteur - 1) PAR PAS DE 1 FAIRE
         ECRIRE(ref_fichier, tab(index))
      FIN POUR
      FERMER(ref_fichier)
      FIN
FIN ALGORITHME
```

Exercice

Écrire l'algorithme de la fonction permettant de calculer la moyenne des salaires de tous les employés pour le mois

Calcul de la moyenne des salaires

```
ALGORITHME moy_salaire
   ENVIRONNEMENT
      entrée :
         matricule_recherche : entier
      sortie :
         réel #la moyenne des salaires
      interne :
         ref_fichier : FICHIER DE Employe
         e : Employe
         compteur : entier
         somme_salaire : réel
```

```
   TRAITEMENT
      DEBUT
         compteur ← 0
         somme_salaire ← 0
         #On commence par ouvrir le fichier
         ref_fichier ← OUVRIR_EN_LECTURE("employe.dat")
         TANT QUE (NON EOF(ref_fichier)) FAIRE
            e ← LIRE(ref_fichier)
            somme_salaire  ← somme_salaire + e.taux_horaire ×
    e.total_heures
            compteur ← compteur + 1
         FIN TANT QUE
         FERMER(ref_fichier)
         #on peut maintenant calculer la moyenne et la renvoyer
         moy_salaire ← somme_salaire / compteur
      FIN
FIN ALGORITHME
```