TP #7 & #9 – APL2

8 et 15 janvier 2020
Erwan Kerrien

Rappel : si vous avez des questions sur ce TP ou sur le cours, n'hésitez pas à m'envoyer un mail à Erwan.Kerrien@inria.fr (je consulte plus rarement mon mail Erwan.Kerrien@univ-lorraine.fr).

Ce TP est prévu sur 4h, et est donc à faire sur les séances des 8 et 15 janvier. L'objectif final est de pouvoir lire et ordonner les informations contenues dans un gros fichier de base de données. La première séance du 8 janvier sera dévolue à la lecture de ce fichier et la mise en place d'une structure permettant de stocker une information atomique (de base). La deuxième séance servira à implanter la structure de données permettant de stocker et manipuler efficacement toutes les informations du fichier. Cette structure de données sera une table de hachage. Le TP 8, intermédiaire, sera une introduction pratique aux tables de hachage. Pour réaliser le TP9, vous aurez besoin d'utiliser ce que vous aurez vu lors du TP8, ainsi que ce que vous avez vu dans le TP3.

La partie 2 concernant le TP7 (lecture du fichier et structure atomique) est à me rendre pour le lundi 13 à 8h.

La partie 3 concernant le TP9 (structure de données et stockage du fichier en entier) est à me rendre pour le vendredi 17 à 20h.

Tout ce qui apparaît sous une section **Exercice** est à me rendre (voir pp. 2 et 3) : vous devez a minima me rendre les fichiers `readCSV.c`, `Element.c` pour le lundi 13 et `statsPrenoms.c` pour le vendredi 17.

Vous pouvez, si vous le souhaitez et si vous le jugez utile, m'envoyer un texte de commentaires. Ne pas le faire ne vous exposera à aucun retrait de points. En revanche, ne pas commenter vos codes d'une manière ou d'une autre sera sanctionné : les commentaires sont notés sur 5 points (sur un total de 20) pour chaque TP.

Les fichiers de réponse (code source, texte...) sont à déposer sur arche.

1 Objectifs du TP

1.1 Le format CSV

Ce TP a pour objectif de lire un gros fichier au format csv (*Comma Separated Values*). Ce format est une manière très courante de stocker une base de données simple dans un seul fichier texte. Simple signifie que les informations sont tabulées : chaque information est un enregistrement qui contient un nombre prédéfini de champs. Les fichiers de ce type peuvent être par exemple importés dans un tableur (libreoffice calc ou excel pour les windowsiens) puisque chaque enregistrement peut être stocké dans une ligne, et chaque champ correspond à une colonne.

Dans le format CSV, chaque ligne du fichier correspond à un enregistrement. La première ligne est spéciale et indique l'en-tête des colonnes (le nom de chaque champ). Dans une ligne, les champs sont séparés par un caractère : le fichier considéré ici utilise le caractère ; et le code demandé ne gérera que ce cas. Un champ est stocké sous forme de chaîne de caractères. Un champ peut être vide.

Le fichier suivant, appelé `Test.csv`, est un exemple de fichier CSV avec ; pour séparateur, comportant 9 enregistrements, chaque enregistrement comportant quatre champs : Nom, Prénom, Âge et Genre. On remarque que le fichier contient 10 lignes : la première ligne indique le nom de chaque champ. De plus, même si l'âge est a priori un nombre, celui-ci est stocké sous forme de chaîne de caractères ("21" pour l'âge 21 ans).

```
Nom;Prénom;Âge;Genre
Enfant;Hélène;44;F
Enfant;Ludivine;47;F
Flaille;Abdel;19;M
Flaille;Akim;23;M
Flaille;Yves;21;M
Neymar;Jean;24;M
Titegoute;Justine;78;F
Yapudebiairedenlefrigo;Robin;67;M
Kerrien;Erwan;;M
```

1.2 Le fichier cible

Le fichier que vous allez traiter est fourni par le site de L'INSEE `https://www.insee.fr`. Vous pouvez le télécharger depuis `https://www.insee.fr/fr/statistiques/fichier/2540004/dpt2018_csv.zip`. Vous en trouverez la description sur `https://www.insee.fr/fr/statistiques/2540004` (onglet Dictionnaire, deuxième fichier). Lisez attentivement cette description qui vous sera utile. **Attention, ce fichier n'a pas les mêmes champs que le fichier `Test.csv` précédent.**

L'objectif du TP est de lire ce fichier et de stocker les informations suivantes pour chaque prénom du fichier : le genre, le nombre total d'occurrences, la première année d'apparition. Si un prénom est utilisé à la fois au féminin et au masculin (par exemple Camille), alors deux enregistrements distincts seront stockés.

2 Lecture de fichier et enregistrement

Exercice : Lecture de fichier CSV

Écrire un fichier `readCSV.c` qui prend en paramètre un fichier et affiche les enregistrements qu'il contient. Par exemple la command `readCSV Test.csv` (voir `Test.csv` ci-dessus) affichera

```
New line:
[0] 'Enfant' [1] 'Hélène' [2] '44' [3] 'F
'
New line:
[0] 'Enfant' [1] 'Ludivine' [2] '47' [3] 'F
'
New line:
[0] 'Flaille' [1] 'Abdel' [2] '19' [3] 'M
'
New line:
[0] 'Flaille' [1] 'Akim' [2] '23' [3] 'M
'
New line:
[0] 'Flaille' [1] 'Yves' [2] '21' [3] 'M
'
New line:
[0] 'Neymar' [1] 'Jean' [2] '24' [3] 'M
'
New line:
[0] 'Titegoute' [1] 'Justine' [2] '78' [3] 'F
'
New line:
[0] 'Yapudebiairedenlefrigo' [1] 'Robin' [2] '67' [3] 'M
'
New line:
[0] 'Kerrien' [1] 'Erwan' [2] '' [3] 'M
'
```

Plutôt que d'utiliser `fscanf`, vous pourrez utiliser à profit les trois fonctions suivantes (dont vous pouvez accéder à l'aide via la commande `man <nom de fonction>`) : `getline`, `strsep` et `feof`.

Bonus : vous pouvez remarquer que le genre se termine par un retour à la ligne. Modifiez votre programme pour qu'il ne s'affiche pas.

Exercice : Structure de stockage d'un enregistrement

Dans ce qui suit, on utilisera les booléens définis par le type `bool` dans `stdbool.h`. Ce fichier définit entre autres les constantes `true` et `false`. On l'emploie en ajoutant la directive `#include <stdbool.h>` en début de fichier source C.

Revenons au gros fichier qui nous intéresse. Nous voulons stocker pour chaque prénom et chaque genre le nombre d'occurrences ainsi que la première année d'attribution du prénom. Après avoir lu la description du fichier (voir ci-dessus), vous répondrez aux questions suivantes (tout sera écrit dans un fichier `Element.c`)

1. parmi les champs d'un enregistrement, lesquels sont intéressants, et lesquels sont inutiles ?
2. proposez une structure C, que vous appellerez `Element`, qui permet de stocker les champs utiles d'un enregistrement. Vous ferez attention au type à donner à chaque champ.
3. écrivez une fonction `bool ElementCompare(Element E1, Element E2)` qui renvoie `true` si les deux éléments ont le même prénom et le même genre et `false` sinon.
4. définissez un `element_invalide` et écrivez une fonction `bool ElementEstValide(Element E)` qui renvoie `true` ou`false` selon que l'élément est valide ou non. On commencera par définir une valeur invalide pour chaque champ de la structure. Puis on pourra se demander quels sont les champs obligatoirement valides pour qu'un enregistrement le soit.
5. écrivez une fonction `ElementAffiche(Element E)` qui affiche un élément. On prendra garde à détecter et afficher correctement un élément invalide.
6. en vous inspirant de ce que vous avez fait pour `readCSV`, écrivez une fonction `Element ElementLire(char *ligne)` qui prend en entrée une ligne lue dans un fichier (par exemple avec `getline`), qui en lit les champs, remplit un `Element` et le renvoie. Cette fonction renvoie `element_invalide` en cas de problème avec le format de la ligne ou avec les données qu'elle contient.

3 Structure de données

Exercice : Table de hachage

En vous inspirant des TP précédents, notamment les TP3 et T8, écrivez les structures et fonctions qui permettent de stocker des `Element` dans une table de hachage. Vous prendrez en compte les éléments suivants :

— La clé sera le prénom, la fonction de hachage sera la somme des codes ascii des lettres de la clé, modulo la taille de la table.

— Les collisions seront gérées par chaînage (autrement dit, chaque case du tableau de la table de hachage contient une liste chaînée d'`Element`)

— lors de l'insertion d'un `Element`, si la clé (prénom) est déjà présente, on mettra à jour l'`Element` stocké en : 1) mettant à jour le nombre d'occurrences (ajout des nouvelles occurrences trouvées) et 2) mettant éventuellement à jour l'année de première apparition (si antérieure à l'année actuellement stockée).

Vous écrirez ensuite un programme `statsPrenoms.c` qui utilise ce code et

1. compte et affiche le nombre de prénoms stockés dans le fichier passé en premier paramètre (obligatoirement donné)
2. affiche l'`Element` correspondant au prénom passé en deuxième paramètre (s'il est donné)
3. calcule et affiche le nombre de prénoms : minimal, maximal et moyen dans les cases de la table de hachage (statistiques sur la longueur des listes chaînées stockées dans la table).

Note : vous pourrez faire varier la taille de la table de hachage. Une bonne manière de voir l'effet de cette taille est d'estimer le temps d'exécution de `statsPrenoms`, ce qui peut se faire en débutant la ligne de commande d'exécution par `time` (par exemple : `time statsPrenoms fichier.csv`).