Langage C

ESILV S5 ⋈ 2017–2018

TD 6 & 7 – **Problème noté**

Attention bien lire cet encadré :

- Ce travail est à réaliser individuellement ou en binôme (même groupe de TD).
- Il est à rendre au plus tard à 22h00 samedi 16 décembre 2017.
- Vous déposerez **sur Moodle un fichier** `.zip`, par binôme, contenant :
  - le **code source** (les fichiers `.c` et `.h`)
  - le **mini-rapport** en `PDF` de 4 pages maximum.
- Un rendu sans le rapport sera noté 00/20.
- Un programme avec des *Warnings* à la compilation sera noté avec un malus pouvant aller jusqu'à -5 points selon la nature du/des *Warnings*.
- Un programme ne compilant pas ou ne s'exécutant pas sera noté 00/20.
- Une triche entraîne une note de 00/20 à l'ensemble du module.

Préambule

Pour gérer un (très) grand nombre de personnes – par exemple un annuaire – il peut paraître judicieux de stocker les informations associées à ces personnes dans un ensemble trié par ordre alphabétique sur les noms. En effet, l'accès en sera plus rapide (par exemple avec un algorithme de recherche par dichotomie).

Cependant, l'alphabet ne disposant que de 26 lettres il y a nécessairement beaucoup de personnes associées à la même / aux mêmes premières lettres... et finalement l'ordre alphabétique n'offre pas une organisation satisfaisante pour un accès rapide aux éléments de l'ensemble.

On se propose donc dans ce double TD d'organiser les éléments au sein d'un ensemble particulier, appelé « **table** » par la suite, où la position d'un élément est fournie par la fonction `index`.

Cette fonction est définie de la manière suivante. En considérant N la taille non nulle de la « table », B un entier non nul strictement positif appelé "base", s une chaîne de caractères, avec $s =$ "$c_0c_1c_2...c_k$" pour $k \geqslant 0$ (où c_i est un caractère), et `ascii` la fonction retournant le code ASCII d'un caractère :

$$index(s, B, N) = \left(\sum_{i=0}^{k} ascii(c_i).B^i \right) \text{ modulo } N$$

✎ Le modulo permet de garantir que la position fournie par la fonction `index` est un entier dans l'intervalle $[0, N[$.
✎ La fonction `index` assure une bonne répartition des éléments lorsque la base B est un nombre premier.

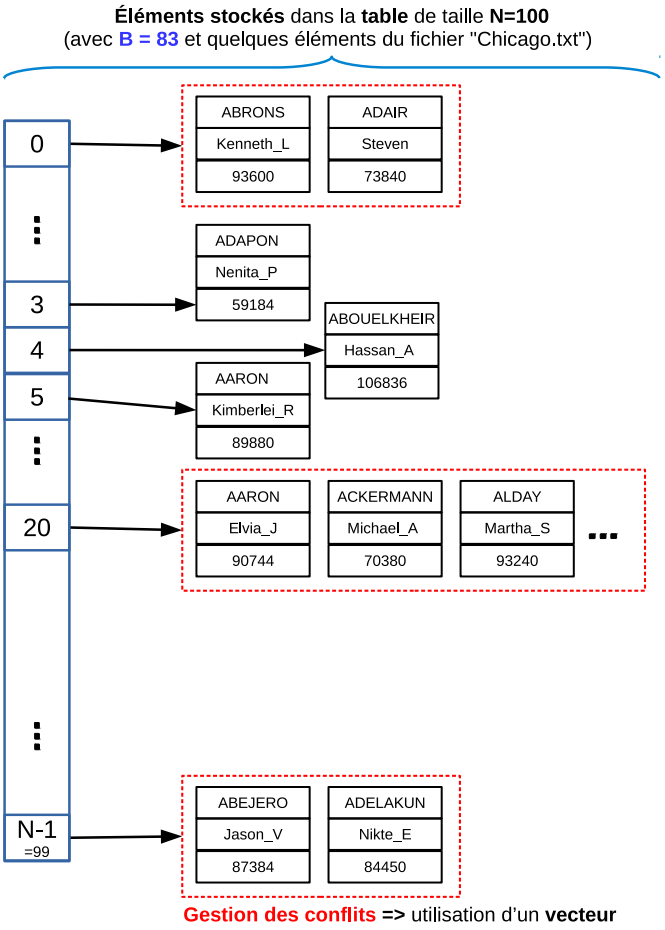
Pour ce double TD, les données manipulées seront celles correspondantes aux **fonctionnaires de la ville de Chicago aux USA** (*cf.* fichier `Chicago.txt`).

Pour les besoins du problème, la chaîne de caractères à considérer pour la fonction `index` sera la concaténation des 4 premières lettres du nom et des 2 premières lettres du prénom d'un fonctionnaire. ⇒ **La fonction `index` aura donc 4 paramètres.**

Par exemple,
`index("AARON","Karina",83,100) = 29`

En cas de conflit(s), c'est à dire lorsque 2 ou plusieurs éléments se retrouvent à une même position, ces éléments sont alors rangés par **ordre alphabétique sur le nom** au sein d'un **vecteur** à ce même index.

✎ Pour faciliter le code, même s'il n'y a qu'un seul élément à un index donné, il sera quand même stocké au sein d'un vecteur à cet index.



1 Énoncé du problème

Créer un programme

— qui commence par demander à l'utilisateur la taille et la base souhaitées pour la « table », puis initialise cette « table » ;
  ↯ il est nécessaire de définir un type structuré adapté pour une telle « table ».

— qui ensuite affiche un **menu** proposant les fonctionnalités suivantes :

1. index
   - l'utilisateur devra fournir un nom et un prénom souhaités
   - le programme affichera l'index associé à ces nom et prénom (indépendamment des éléments présents dans la « table »)
2. ajouter
   - l'utilisateur devra fournir le nom, le prénom et le salaire du fonctionnaire à ajouter
   - le programme ajoutera ce fonctionnaire dans la « table », sauf si déjà présent. Un message indiquera si l'ajout a pu se faire ou non.

3. charger
   - l'utilisateur devra fournir le nombre de fonctionnaires à charger (depuis le fichier `Chicago.txt`)
   - le programme ajoutera – sans doublon – les fonctionnaires demandés dans la « table ».
4. afficher salaire
   - l'utilisateur devra fournir le nom et le prénom du fonctionnaire souhaité
   - le programme affichera le salaire du fonctionnaire s'il est présent dans la « table », sinon -1 sera affiché.
5. afficher entre
   - l'utilisateur devra fournir l'index de début et de fin (inclus) souhaités
   - le programme affichera, par index (à afficher) et par ordre alphabétique sur les noms, les données relatives aux fonctionnaires présents dans la « table » entre les index demandés.
6. nombre de conflits
   - l'utilisateur n'a rien à fournir
   - le programme affichera le nombre de fois où il y a un conflit dans le stockage des fonctionnaires dans la « table ».
7. taille moyenne des conflits
   - l'utilisateur n'a rien à fournir
   - le programme affichera le nombre moyen d'éléments à un même index lorsqu'il y a un conflit dans la « table ».
8. supprimer
   - l'utilisateur devra fournir le nom et le prénom du fonctionnaire à supprimer
   - le programme supprimera ce fonctionnaire dans la « table », sauf si non présent. Un message indiquera si la suppression a pu se faire ou non.
9. supprimer entre
   - l'utilisateur devra fournir l'index de début et de fin (inclus) souhaités
   - le programme supprimera les fonctionnaires présents dans la « table » entre les index demandés.
10. quitter
    - l'utilisateur n'a rien à fournir
    - le programme s'arrêtera.

Sauf pour le dernier item du menu, celui-ci sera de nouveau affiché à l'utilisateur pour qu'il puisse faire une nouvelle action de son choix.

1.1 Le fichier `Chicago.txt`

Le fichier `Chicago.txt` est formaté de la manière suivante :

— la première ligne contient un unique entier indiquant le nombre de lignes suivantes dans le fichier.

— chaque ligne suivante correspond à un fonctionnaire, elle est composée

— du nom du fonctionnaire, d'un espace, du prénom, d'un espace, et du salaire annuel en USD (nombre entier).

✎ Le nom et le prénom ne contiennent donc pas d'espace (le cas échéant le caractère '_' est utilisé dans le fichier)

À noter que les données dans ce fichier ne sont pas triées.

1.2 Contraintes

Une attention particulière devra être portée sur la qualité du code :
- découpage en (sous-)fonctions [1].
- noms des variables et fonctions explicites
- organisation adaptée du code (fichiers `.c` / `.h`)
- commentaires à bon escient

Seules [2] les bibliothèques suivantes peuvent être utilisées : `<stdlib.h>`, `<stdbool.h>`, `<string.h>` et `<stdio.h>`.

2 Mini-rapport

Un mini-rapport de 4 pages maximum est à rendre avec le code.

Doivent y figurer :
- les nom et prénom des élèves qui ont fait le travail ;
- une explication en français du type choisi pour manipuler la « table » ;
- une explication en français de comment est codée chaque fonctionnalité proposée dans le menu (en indiquant le numéro de la fonctionnalité dans le menu). Un petit schéma peut éventuellement venir compléter l'explication.

1. donc pas tout dans le `main`!!!
2. mais pas `<math.h>`