MPI Mini Projet Info

Déroulement par élève :

4 séances de 4h encadrées + autant de temps en travail personnel
Réalisation du projet par groupe de **2 (**soit 64 h de réalisation par binôme)

Objectif de ce TP de synthèse:

Réaliser un travail sérieux d'analyse d'un sujet de SDA (papier, crayon)
Faire découvrir aux élèves un autre langage (Ada)en appliquant les méthodes vues en SDA1.
Développer l'autonomie (auto-apprentissage)
Initier au travail en groupe (2 élèves)

Rôle des enseignants :

Apporter une aide méthodologique, répondre aux questions, aiguiller les recherches.

Déroulement:

Séance 1: travail d'analyse du sujet situé plus bas. (jusqu'à validation définitive de l'enseignant)
Pas de programmation sans cette validation.
La structure de donnée et l'algorithme général doivent être réalisés.

Séance 2: fin du travail d'analyse si besoin (jusqu'à validation définitive de l'enseignant)
Début de la programmation en Ada

Séance 3 et 4 : suite de la programmation en Ada jusqu'à présentation d'une application conviviale, sécurisée et rigoureuse.

Notation:

Elle portera sur le **compte rendu final** (CR personnel + Listing) mais tiendra compte aussi du rôle actif de chaque groupe pendant les séances.
Remise de votre compte rendu final à **S Collart** : 3 jours après votre dernière séance soit :

- Avant le Je 29/3 inclus pour le groupe A
- Avant le Lu 2/4 inclus pour le groupe AB
- Avant le Je 29/3 inclus pour le groupe B

Le sujet

Réalisation d'un mini moteur de recherche

Cahier des charges

Présentation

La société ***Ch'tite SSII*** souhaite mettre en ligne (en interne) son propre moteur de recherche. Celui-ci doit permettre de retrouver rapidement quels sont les fichiers (quelles lignes et quelles colonnes) contenant un mot saisi par l'utilisateur. Par comparaison, un tel moteur ressemble à ceux utilisés sur Internet. Bien entendu, celui qui sera développé pour ***Ch'tite SSII*** sera beaucoup moins évolué que ces derniers.
Ses principales restrictions sont les suivantes :
- Aucune aspect réseau
- Manipulation de fichiers texte exclusivement (situés dans le répertoire contenant votre programme)
- limitation à deux fichiers txt : fichier1.txt et fichier2.txt
- Pas de prise en compte des déclinaisons des mots. Autrement dit, les mots «élève » et « élèves » par exemple seront traités comme des mots différents.
- Recherche limitée à un seul mot à la fois
- Dans un premier temps, fichiers ne contenant aucune ponctuation, aucun caractère accentué comme é,è,ê,ë,à,ç …

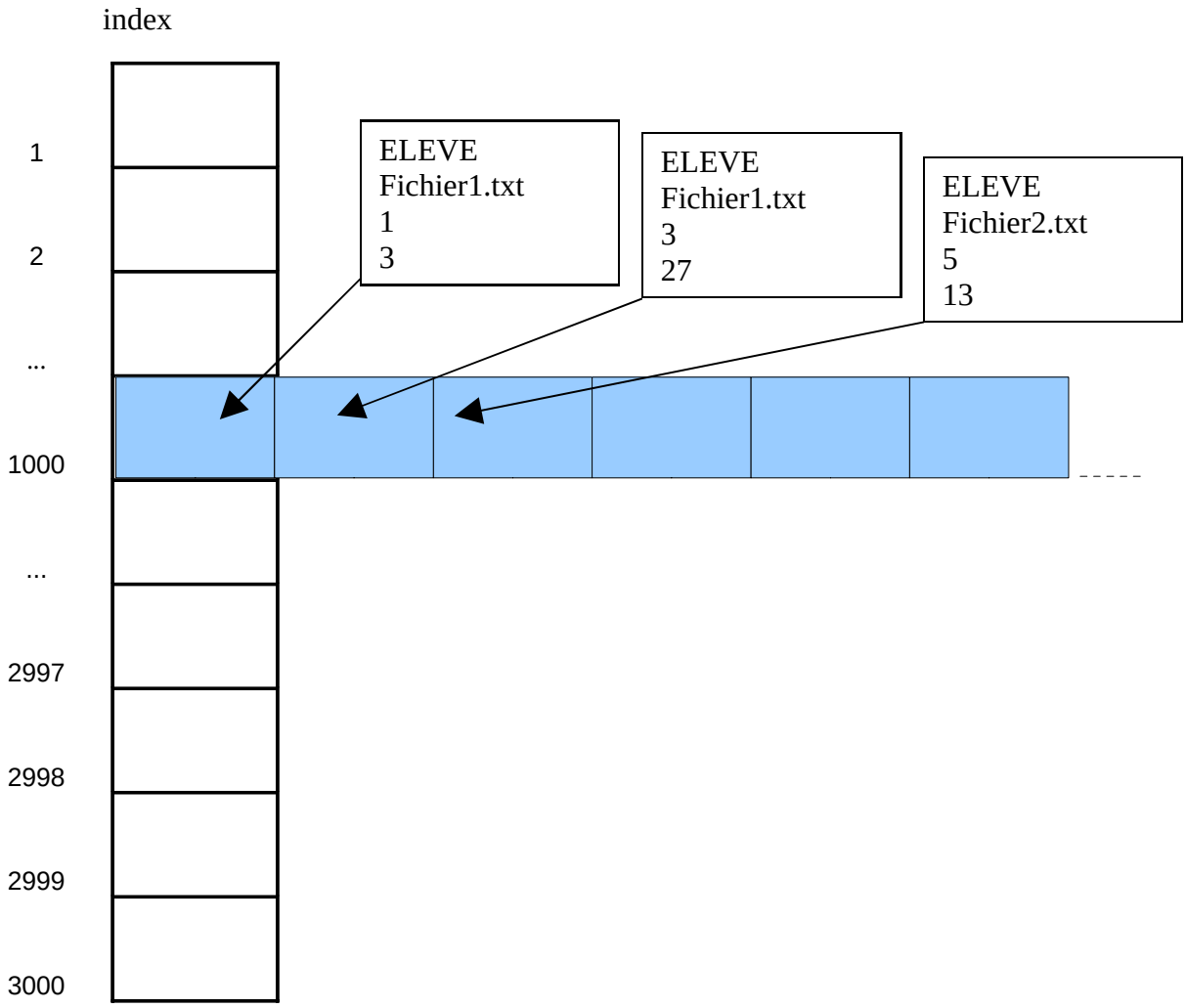
Votre binôme figure sur la liste des binômes retenus pour le développement de ce moteur. Le meilleur des moteurs verra son binôme obtenir une récompense conséquente.

Première analyse
La réalisation du moteur n'est pas très difficile. Les structures de données mises en œuvre sont assez simples et seront élaborées ci-après. De plus, il n'y aucune difficulté algorithmique majeure.

La structure de donnée majeure sera un tableau nommé « **index** ». Il s'agit d'un tableau de taille **TAILLE** (avec **TAILLE** égale à 3000 par exemple) dont les éléments sont des tableaux de taille **TAILLE2** de structures secondaires.

Chaque structure secondaire se compose :

- d'une chaîne de caractères (en majuscule) nommée **MOT** qui correspond à un mot trouvé dans un ou plusieurs fichiers
- d'une chaîne de caractères nommée **NOMFICHIER** qui correspond au nom du fichier dans lequel le **MOT** a été trouvé
- Une entier nommé **ligne** désignant la ligne où ce **MOT** a été trouvé dans le fichier nommé **NOMFICHIER**
- Une entier nommé **colonne** désignant la colonne où ce **MOT** a été trouvé dans le fichier nommé **NOMFICHIER**



La figure précédente montre une partie de l'index obtenu à partir de mots contenus dans deux fichiers texte « fichier1.txt » et « fichier2.txt ».
Le mot ELEVE (pouvant s'écrire élève ou elève ou éleve ou Elève ou ELEVE ou …) est présent à 2 reprises dans le fichier « fichier1.txt » et une seule fois dans le fichier « fichier2.txt »
On suppose ici que le **HASHCODE** de ce mot vaut 1000. Mais qu'est ce qu'un **HASHCODE** ????

Afin de ranger efficacement et d'accélérer la recherche d'un mot dans l'index, nous allons utiliser une fonction de hachage. Une telle fonction prend une chaîne de caractères en paramètre et produit une valeur entière comprise entre 0 et TAILLE-1, où **TAILLE** est la taille de l'index (le tableau). Le recours à une telle fonction permet d'obtenir directement l'indice de l'élément du tableau correspondant à un mot. Ainsi, en considérant l'exemple représenté ci-dessus, on peut comprendre que la fonction de hachage a retourné respectivement la valeur 1000 pour le mot «ELEVE».

Voici la définition de la fonction de hachage :

```
Fonction hash(pointeur sur une chaine de caractères nommée mot) retourne un
entier
h:entier;
début
Pour chacune des lettres de mot faire
h reçoit le  Reste de la division de (64*h + code ascii de la lettre en
question) par TAILLE;
FinPour
Retourne h;
Fin
```

Notez qu'il faut prendre un TAILLE assez grand (3000 par exemple) de façon à éviter que la fonction de hachage retourne la même valeur pour deux mots différents.

Options et évolutions possibles :

Il est demandé à tous de réaliser un projet moteur comme indiqué ci-dessus et d'y inclure au moins l'une des options suivantes :

1. Demander à l'utilisateur combien de fichiers txt, il souhaite examiner. Saisir alors les noms de fichiers au sein d'un tableau pour ensuite effectuer le travail du moteur sur l'ensemble des fichiers
2. Traiter le cas des fichiers ponctués.
3. Traiter le cas des fichiers avec caractères accentués.
4. ?
5. ?
6. ?
7. ?

Les plus rapides pourrront traiter plusieurs ou la totalité des options.