Mathieu Martinez Clarisse Tarrou

Projet Saisie Intuitive

Rapport

1 STRUCTURES DE DONNEES UTILISEES

- Dictionnaire : structure utilisée pour construire l'arbre représentant le dictionnaire issu de « dictionnaire.txt ».

```
struct Dictionnaire
{
   char data;

   unsigned fin_de_mot: 1;

   struct Dictionnaire *gauche, *suite_mot, *droite;     //suite_mot est le suivant
};
```

data : une des lettres du mot

fin_de_mot : entier valant 1 si la lettre correspondante est la dernière d'un mot, 0 sinon.

gauche : renvoie sur une autre lettre de même rang (c'est-à-dire qui a la même place dans le mot) classée avant dans l'alphabet. Par exemple si on veut distinguer « abasourdi » de « abeille »(distinction qui se fait au niveau de la troisième lettre), le « a » de « abasourdi » sera obtenu par des renvois à « gauche » successifs à partir du « e » de « abeille ».

Même raisonnement pour « droite » mais les lettres sont parcourues dans l'ordre alphabétique.

suite_mot : pointe vers la lettre suivante dans le mot (par exemple passage de la troisième à la quatrième lettre d'un mot).

- Mots_courant : structure utilisée pour fluidifier l'ajout et la suppression de mots dans le dictionnaire des mots courants, en permettant de générer une liste chainée composée de tous les mots de ce dictionnaire dans l'ordre alphabétique. Est également utilisée pour générer le fichier texte correspondant à cette liste : « mot_courants_optimized.txt » à partir du fichier « mots_courants.txt ».

```
typedef struct mot_courant{ //Struture permettant de manipuler facilement les mots courants et de les insérer facilement dans le fichier.

   char mot[50];   //Le mot le plus long  en français fait 25 lettres ; le mot le plus long du monde fait 136 lettres et le deuxième plus long 63 lettres
   int occurence; //Nombre d'occurrences du mot.
   struct mot_courant *suivant;
}Mots_courant;
```

mot : stocke le mot à ajouter

occurence : occurrence du mot en question

suivant : mot suivant dans le dictionnaire des mots courants trié par ordre alphabétique

- Dictionnaire_courant : structure utilisée pour construire l'arbre représentant le dictionnaire issu de « mots_courants_optimized.txt ».

```
struct Dictionnaire_courant //Structure d'arbre des mots courant
{
   char data;

   unsigned fin_de_mot: 1;
   int occurence;
   struct Dictionnaire_courant *gauche, *suite_mot, *droite;     //suite_mot est le suivant
};
```

Les entrées « data », « fin_de_mot », « gauche », « suite_mot » et « droite » ont la même utilité que dans la structure « Dictionnaire ».

occurence : occurrence du mot ; implémenté sur les nœuds correspondant à la dernière lettre du mot en question.

2 ALGORITHMES PROPOSES

2.1 IMPLEMENTATION DES ARBRES A PARTIR DES FICHIERS .TXT

Les arbres correspondants à chacun des dictionnaires sont définis à partir d'une racine « root », et de nœuds « Dictionnaire» pour l'arbre défini à partir de « dictionnaire.txt » et « Mots_courants » pour l'arbre défini à partir de «mots_courants.txt » . Le niveau de parenté d'un élément de l'arbre correspond à la place de la lettre définie par le nœud « Node » dans le mot en cours de représentation. Chaque mot est ainsi représenté par une suite de nœuds tous parents les uns des autres. La fin d'un mot est indiquée par l'entier « fin_de_mot », qui vaut 1 si le nœud en question représente la fin d'un mot, 0 sinon. Par exemple, les mots « abeille » et « abeilles » seront représentées par la même liste chaînée, plus ou moins le « s » : les entiers « fin_de_mot » du « e » final et du « s » vaudront donc tous les deux 1.

Lettre	**A**	**B**	**E**	**I**	**L**	**L**	**E**	**S**
fin_de_mot	0	0	0	0	0	0	1	1

2.2 AJOUT/SUPPRESSION D'UN MOT DANS UN ARBRE

Pour ajouter un mot dans un arbre, on cherche lettre par lettre les correspondances au début du mot. Par exemple dans le cas d'« abeille » : a -> b -> e -> … où « -> » est le nœud suivant (« eq » dans notre structure « Node »). S'il apparaît qu'un de ces liens n'existe pas, par exemple qu'il n'existe au moment de l'ajout aucun mot commençant par « abei », on ajoute les lettres manquantes une à une jusqu'à former le mot. Si le mot à ajouter est déjà contenu dans un autre mot (i.e. on veut ajouter « abeille » alors que le dictionnaire des mots courants contient déjà le mot « abeilles »), on fait passer l'entier « fin_de_mot » à 1 au nœud correspondant.

Dans ce même cas, supprimer un mot de l'arbre revient à faire passer l'entier « fin_de_mot» de la dernière lettre du mot à supprimer à 0. Dans le cas où une partie des lettres seulement devraient être supprimées, on supprime la lettre du chaînage en faisant pointer son nœud antérieur vers son nœud postérieur.

2.3 RECHERCHE D'UN MOT

La recherche d'un mot se fait grâce à un algorithme de parcours en profondeur du dictionnaire courant. Les mots commençant par les lettres saisies par l'utilisateur sont rangés dans une liste chaînée, puis une fonction de recherche de maximum permet de renvoyer les trois éléments d'de ce tableau sont enfin renvoyés à l'utilisateur via la console. L'utilisateur a alors la possibilité de choisir l'un de ces mots en appuyant sur les touches « 1 », « 2 » ou « 3 ». La sélection d'un mot permet d'ajuster son « occurence » en lui ajoutant 1.

Dans le cas où moins de trois mots correspondants seraient disponibles dans le dictionnaire des mots courants, les mots manquants sont recherchés de la même manière dans le dictionnaire « dictionnaire.txt ». Dans le cas où le mot sélectionné par l'utilisateur correspondrait à un mot provenant de ce dictionnaire, le mot en question serait ajouté au dictionnaire des mots courants avec une occurrence de 1.

3 DIFFICULTES RENCONTREES ET LIMITES DU PROJET

La phase la plus complexe et la plus chronophage de notre projet à été le parcours récursif de l'arbre.

Nous n'avons pas modélisé le cas où le mot recherché par l'utilisateur ne ferait pas partie des trois mots renvoyés par la fonction de recherche, et nous sommes limités à renvoyer les trois mots dont les occurrences sont les plus élevées et qui commencent par la même chaîne de caractères que celle saisie par l'utilisateur. Nous n'avons également géré le cas des lettres avec accent : « à », « â », « é », « è », …

4 REPARTITION DU TRAVAIL

Si la réflexion initiale à été commune, le travail a été réparti entre les deux membres du binôme comme suit :

- Mathieu s'est occupé de la gestion des arbres : création des arbres, parcours, ajout de nœuds, …
- Clarisse s'est occupée de la gestion des listes : mot_courant, ajout/suppression dans un fichier, menu, interface utilisateur.