NF16 - TP 4 – Les ABR

Indexation d'un texte

Introduction

Nous allons utiliser les propriétés des arbres binaires de recherche pour implémenter un exemple d'indexation et de recherche sur un fichier contenant un texte quelconque.

L'arbre binaire de recherche contiendra tous les mots présents dans le texte à indexer :

- chaque nœud de l'arbre contient un mot, ainsi que la liste des positions du mot dans le texte
- une position correspond à un numéro de ligne, un ordre dans la ligne (1 pour le premier mot de la ligne, 2 pour le deuxième mot, etc...), un numéro de phrase
- le texte ne contient que des majuscules, des minuscules et des points (séparateur de phrases)

A. Structures de données

Implémenter les structures de données suivantes :

- la structure **`struct position`** (dont on définira le type synonyme **`Position`**) qui comporte les champs :
  - **`numero_ligne`** de type **`int`**
  - **`ordre`** de type **`int`**
  - **`numero_phrase`** de type **`int`**
  - **`suivant`** de type **`Position*`**
- la structure **`struct listePosition`** (dont on définira le type synonyme **`ListePosition`**) qui comporte les champs :
  - **`debut`** de type **`Position*`**
  - **`nb_elements`** de type **`int`**
- la structure **`struct noeudABR`** (dont on définira le type synonyme **`NoeudABR`**) qui comporte les champs :
  - **`mot`** de type **`char*`**
  - **`positions`** de type **`ListePosition`**
  - **`filsGauche`** de type **`NoeudABR`***
  - **`filsDroit`** de type **`NoeudABR`***
- la structure **`struct arbreBR`** (dont on définira le type synonyme **`ArbreBR`**) qui comporte le champ :
  - **`racine`** de type **`NoeudABR`***
  - **`nb_mots_differents`** de type **`int`**
  - **`nb_mots_total`** de type **`int`**

B. Fonctions de base à implémenter

1. Fonction qui crée une liste de positions vide (renvoie NULL si échec):
   `ListePosition *creer_liste_positions()`

2. Fonction qui ajoute un élément dans une liste de positions en respectant l'ordre du mot dans le texte (renvoie 1 en cas de succès, 0 sinon):
   `int ajouter_position(ListePosition *listeP, int ligne, int ordre, int num_phrase)`

3. Fonction qui crée un ABR vide (`renvoie NULL en cas d'échec`) :
   `ArbreBR *creer_abr()`

4. Fonction qui ajoute un nœud dans l'arbre en respectant les règles d'insertion dans les ABR (renvoie 1 en cas de succès, 0 sinon):
   `int ajouter_noeud(ArbreBR *arbre, NoeudABR *noeud)`

   On se basera sur l'ordre lexicographique pour déterminer la position d'un nœud dans l'arbre :
   avion < bagage < bateau < cabane < cabine <voiture=VoiTure

5. Fonction qui lit un fichier et ajoute tous les mots dans un ABR (renvoie le nombre de mots lus) :
   `int charger_fichier(ArbreBR *arbre, char *filename)`

6. Fonction qui recherche un nœud dans un ABR (renvoie NULL si nœud non trouvé) :
   `NoeudABR *rechercher_noeud(ArbreBR *arbre, char *mot)`

7. Procédure qui affiche la liste des mots classée par ordre alphabétique (choisissez le bon type de parcours des nœuds de l'arbre) :
   `void afficher_arbre(ArbreBR arbre)`

C. Programme Principal :

Programmer un menu qui propose les fonctionnalités suivantes :

1. **Créer un ABR.**
2. **Charger un fichier dans l'ABR** : Afficher le nombre de mots lus dans le fichier.
3. **Caractéristiques de l'ABR** : Afficher le nombre de nœuds, la profondeur, et si l'ABR est équilibré.
4. **Afficher tous les mots distincts par ordre alphabétique**
5. **Rechercher un mot** : Afficher le n° de ligne, l'ordre dans la ligne, le n° de phrase pour chaque occurence.
6. **Afficher les phrases contenant deux mots** : Afficher toutes les phrases contenant les deux mots saisis.
7. **Quitter : La** mémoire allouée dynamiquement doit être libérée.

Bonus. **Equilibrer l'ABR :** Proposer à l'utilisateur d'équilibrer l'arbre si nécessaire

NF16 - TP 4 – Les ABR

Consignes générales :

L'organisation du projet sera la suivante :

- Fichiers d'en-têtes **`arbre.h`**, **`liste.h`** et **`outils.h`** pour les déclarations des structures et fonctions
- Fichiers source **`arbre.c`**, **`liste.c`** et **`outils.c`** contenant la définition de chaque fonction
- Fichier source **`main.c`** contenant le programme principal.

Votre rapport (deux à quatre pages) devra contenir :

- le rappel des algorithmes vus en cours et utilisés dans le programme
- une description des fonctions et structures de données implémentées que vous avez choisi d'ajouter à celles qui demandées dans cet énoncé
- les difficultés rencontrées et vos choix pour y remédier