BOUZAR Massil
GAJAN Antoine UV : NF16
30/03/2022

NF16 - TP3 - RAPPORT

Ce rapport contiendra :

- Description des structures
- Complexité des fonctions
- Ajout / Modification du sujet (prototypes, fonctions,...)
- Correctifs apportés suite à la démonstration réalisée en TP

Introduction

Ce TP a pour objectif de familiariser les étudiants avec les notions de structures de données et de pointeurs. Ainsi, dans le cadre de l'élection présidentielle 2022, nous souhaitons utiliser une liste linéaire simplement chaînée pour modéliser la liste des électeurs.
Pour chaque électeur, on prendra en considération son nom, son numéro de carte d'identité ainsi que son choix de vote. On définira ensuite le type T_Electeur comme un pointeur sur un électeur.

Description des structures

Le type "électeur" a été implémenté grâce à un typedef pour représenter un électeur. Un électeur est modélisé par son nom, son numéro d'électeur cin_num, son choix de type entier et d'un pointeur sur l'électeur suivant. Le nom est représenté par un char * car on ne connaît par à l'avance la taille du nom de l'électeur. Le type T_Electeur a été quant à lui créé comme un pointeur sur un électeur.

Complexité des fonctions

★ **T_Electeur creationelecteur() : fonction qui permet de créer un électeur**

Cette fonction réalise une allocation dynamique de mémoire pour un électeur, effectue une opération logique et une affectation. Ces opérations se faisant en temps constant, la fonction est en O(1).

★ **void afficheliste(T_Electeur liste) : fonction qui affiche la liste des électeurs**

La fonction est formée d'une boucle while qui parcourt l'ensemble des électeurs dans la liste. Si on note n le nombre d'électeurs dans la liste, la fonction est en O(n).

- ★ **void ajoutelecteur(T_Electeur *liste, char nom[], long cin_num, int choix) : fonction qui permet d'ajouter un électeur par ordre alphabétique de son nom dans la liste**

La fonction a recours à des affectations, opérations logiques et arithmétiques, et fait appel à creationelecteur() qui sont en O(1). La fonction possède une boucle while qui parcourt les la liste pour ajouter l'électeur au bon endroit en fonction de son nom. Dans le meilleur des cas, l'électeur sera le "premier" dans la liste (premier dans l'ordre alphabétique), donc la fonction sera en Ω(1). Dans le pire des cas, l'électeur aura un nom qui sera le dernier dans l'ordre alphabétique, ainsi on parcourra les n électeurs. La fonction sera en O(n).

- ★ **int comptelecteur(T_Electeur liste) : fonction qui renvoie le nombre des électeurs de la liste**

La fonction parcourt l'ensemble des électeurs et incrémente le compteur à chaque électeur. Si on note n le nombre d'électeurs, la fonction est donc en O(n).

- ★ **int trouvelecteur(T_Electeur liste, long cin_num) : fonction qui recherche un électeur avec le numéro de sa carte d'identité et affiche toutes ses données**

La fonction est composée d'une boucle while qui parcourt les électeurs jusqu'à trouvé celui qui possède le cin_num entré en paramètre. Dans le pire des cas, l'électeur n'est pas dans la liste ou est le dernier de la liste, on a donc parcouru l'ensemble de la liste (composée de n électeurs), donc la fonction est en O(n). Dans le meilleur des cas, ce sera le premier électeur de la liste, la fonction sera en Ω(1).

- ★ **void Supprimelecteur(T_Electeur *liste,long cin_num) : fonction qui permet de supprimer un électeur à l'aide de son numéro CIN de n'importe quelle position de la liste**

A nouveau, cette fonction parcourt l'ensemble des électeurs à l'aide d'une boucle while. Dans le pire des cas, on ne trouve pas l'électeur ou il est le dernier de la liste, la fonction sera en O(n) avec n le nombre d'électeurs de la liste. Dans le meilleur des cas, on supprime le premier électeur de la liste, la fonction sera en Ω(1).

- ★ **void decoupeliste(T_Electeur liste,T_Electeur *gauche,T_Electeur *droite,T_Electeur *blanc) : fonction qui décompose la liste des électeurs en trois sous-listes gauche, droite et blanc selon le choix de candidats**

Pour chacun des n électeurs de la liste de départ, on l'ajoute en fonction de son choix dans la liste gauche, droite ou blanche en faisant appel à ajoutelecteur() qui est en O(n). La fonction sera donc en $O(n^2)$.

- ★ **void triliste(T_Electeur *liste) : fonction qui trie selon le numéro de la carte d'identité**

Cette fonction correspond à un tri à bulles. Le pire cas est atteint lorsque l'électeur avec le plus petit cin_num est en fin de liste. Si on note n le nombre d'électeurs, on obtient alors que la fonction est en $O(n^2)$.

- ★ **T_Electeur fusionlistes(T_Electeur gauche, T_Electeur droite) : fonction qui crée une nouvelle liste en fusionnant seulement les deux sous-listes de gauche et de droite**

Soit n le nombre d'électeurs de la liste gauche et m le nombre d'électeurs de la liste droite. On parcourt l'ensemble des n électeurs de la liste de gauche (O(n)) pour ensuite faire pointer le dernier électeur de gauche vers le premier de droite. Puis on trie la liste fusionnée avec la fonction triliste() qui est en $O((n+m)^2)$. La fonction est donc en $O((n+m)^2)$.

- ★ **int compteGD(T_Electeur liste) : fonction qui retourne le nombre d'électeurs de gauche dans la liste**

Cette fonction est similaire à la fonction comptelecteur(). En effet, on parcourt les n électeurs de la liste et on incrémente la variable compteur si le choix de l'utilisateur est 1 ou 3. Cette opération logique se faisant en temps constant, la fonction est en O(n).

- ★ **void libereliste(T_Electeur liste) : fonction pour libérer une liste**

Cette fonction parcourt les n électeurs de la liste et libère l'espace mémoire qui leur était alloué. La libération de mémoire se faisant en temps constant, la fonction est en O(n).

Ajout / modification du sujet (prototypes, fonctions,...)

Nous avons choisi de modifier le prototype de la fonction triliste.
Initiallement, nous avions : void triliste(T_Electeur liste).
Nous l'avons remplacé par void triliste(T_Electeur *liste).

Nous avons opté pour cette modification car lors du tri, on peut être amené à changer le premier élément de la liste, et donc la valeur de la variable liste. Si l'on n'utilise pas la notion de pointeurs, une copie de la variable liste sera effectuée. On pourra donc modifier tous les électeurs sauf celui en première position.

Correctifs apportés suite à la démonstration réalisée en cours

Suite à la démonstration en cours, nous nous sommes aperçus que la fonction scanf("%s", nom); ne nous permettait pas de stocker un nom contenant des espaces. Afin

de remédier à ce problème, nous avons cherché sur Internet les différentes solutions possibles et nous avons choisi d'utiliser la fonction scanf comme suit : scanf("%[^\n]s", nom). Elle permet de stocker dans nom l'ensemble des caractères insérés avant le '\n'.

Egalement, dans le choix numéro 6 du menu (découpe et tri des listes), nous avions inversé les commandes affichage de la liste et tri de la liste. Nous avions fait l'affichage des listes avant le tri pour les listes gauche et droite, mais pas pour les listes blanc et fusion. De ce fait, l'affichage pour les listes gauche et droite était incorrect par rapport à la consigne puisque pour les listes gauche et droite, les électeurs n'étaient pas triés selon leur numéro d'électeur. Nous avons inversé ces deux lignes de commande afin de répondre au sujet.