BUCH Warren

Wu Florian

Projet de SDA 2

Sujet : Arbre généalogique

Pour notre projet de SDA 2 sur les arbres généalogiques, nous avons décidé de choisir d’implémenter chaque fonction à notre manière et nous allons vous montrer pour chaque fonction le modèle.

SOMMAIRE :

Fonctions :

```
ident plus_ancien(Genealogie *g, ident x);

void affiche_parente(Genealogie *g, ident x);

void affiche_descendance(Genealogie *g, ident x);

void genealogieFusion(Genealogie *gres, Genealogie *a1, Genealogie *a2);
```

- La fonction genealogieInit :

Elle permet d’initialiser l’arbre généalogique de base, on la code en initialisant avec un choix à 100 pour tab.

Avec un ID de 0 qu’on incrémente

- La fonction genealogieFree :

Fonction qui va détruire l’arbre généalogique, à l’aide d’un free, afin de libérer de la mémoire.

- La fonction adj :

La fonction la plus importante qui permet d’ajouter un individu dans notre arbre.

On l’implémente en créant un nouvel individu et on définit toutes ces père,mere,naissance,décès,nom

On vérifie s'il y a encore assez de place dans tab. Si jamais ça n’est pas le cas, on réalloue de la mémoire dynamiquement.

Puis on doit mettre faine et cadet à jour. Pour cela, on teste d’abord si les parents de l’individu que l’on insère existe. Si c’est le cas, on compare leur fils aîné au nouvel individu que l’on insère.

Si le fils aîné des parents est plus vieux alors on fait une boucle où on compare les dates d’un frère et de son suivant avec celle de l’individu que l’on veut insérer jusqu’à que l’on trouve où insérer l’individu.

Sinon, le nouvel individu devient le fils aîné des parents et l’ancien individu le cadet de l’individu que l’on vient d’insérer.

- Fonction compDate :

Compare 2 dates de naissance et retourne 1 si x est plus grand que y 0 si y est plus grand et 2 si les dates sont identiques.

Elle nous servira pour les fonctions où on cherche une personne.

- Fonction *get :

Fonction pour avoir les informations sur l'individu à l'aide de son identifiant. Sa place dans le tableau est tout simplement son identifiant – 1.

- Fonctions bool (freres_sœurs/ cousins/ fils) :

Fonctions pour vérifier s'ils sont frères/cousins/ etc …

Pour freres_soeurs

On prend le parent des deux individus et on compare si ils sont identiques

Pour cousins

On prend le parent des deux individus et on compare si ils sont frères et soeurs

Pour fils

On compare juste avec les parents de y le x.

- Fonctions bool (ancêtre et ancetreCommun) :

<u>Pour la fonction ancêtre :</u>

On fait une fonction récursive :

si y == 0 alors on retourne faux

si x == y alors on retourne vrai

sinon on rappelle la fonction ancêtre en remplaçant y par la mère et par le père avec un opérateur OU

- Fonctions d'affichage (freres_sœurs/ enfants/ cousins/ oncles) :

Ils permettent d'afficher le prénom selon la fonction qu'on utilise.

Pour les fonctions d’affichage on vérifie juste les conditions si l’identifiant n’est pas 0 et l’arbre non vide puis on parcourt tout le tableau tab en itératif et on teste ce que l’on veut sur chacun des éléments du tableau.

- Fonctions d’affichage (parente et descendance) :

Ces fonctions affichent la liste de parents ou des enfants dans la descendance de X.

Pour les implémenter on fait un parcours d’arbre récursive et on utilise des fonctions auxiliaires.

Par exemple pour la descendance on affiche les premiers enfants puis à partir de l’ainée on vérifie s’il a un cadet. Puis on rappelle cette fonction auxiliaire afin de d’afficher la 2e générations des enfants.

De même avec parente SAUF qu’il y a plus de conditions à vérifier.

- Fonction généalogieFusion :

Permet de Fusionner deux arbres entre eux.

//fonction incomplète

Dans le principe, si a1 et a2 sont non nuls, il faudrait parcourir tout le tableau de a1 et de a2 et comparer les éléments pour éviter les doublons.

Difficultés rencontrées :

Le premier pas était tout d’abord de comprendre la structure et puis de les implémentés correctement de manière fonctionnelle et cohérente.

A savoir que le parcours de la structure est différent avec ceux vu classiquement en cours. Ici on utilise un ‘get’.

Une des difficultés rencontrée est que nous avions du mal à comprendre comment mettre les champs faine et cadet à jour. En fait, faine pointe vers le fils aîné et les champs cadet pointent vers le frère plus jeune suivant, à la manière d’une liste chaînée.

La plus grosse difficulté que l’on a rencontré est la fonction de fusion, nous avons eu bcp de mal à la faire et elle est d’ailleurs toujours incomplète...