Ludovic 1AG8

CASTIGLIA

Compte rendu de la SAE 1.02

Les structures :

Ce projet est composé de deux structures principales dans lesquelles sont englobées les autres :

tiut un tableau alloué dynamiquement de pointer vers une structure (VilleIUT). Cette dernière stock le nom de la ville dans la variable ville et une liste chainée de maillonDept. Les maillonDept stock le nom du département dans le champ nom, le nombre de place dans le champ nbp et le nom du responsable dans le champ resp. Enfin, les maillonDept possède un pointeur vers le prochain maillon (suiv) et une liste chainée de demande. Une demande stock un nom, le numéro de candidat dans numc, la décision du département dans desdp et la validation du candidat dans valcan ainsi qu'un pointeur vers le prochain maillon.

La deuxième structure est tc, également un tableau alloué dynamiquement de pointer vers une structure Candidat. Un Candidat stock le nom et prénom du candidat, ainsi que son numéro de candidat dans numc, ces 4 notes et le nombre de choix qu'il a fait. Enfin, Candidat possède une liste chainée de type maillonChoix. Ces maillons, en plus de posséder un pointeur vers le suivant, stock le nom de la ville et du département dans respectivement, ville et dep. Ainsi que la décision du département et la validation du candidat.

Les algorithmes :

Les algorithmes de recherche :

```
int recherche_ville(VilleIUT **tiut,char ville[31],int *trouve, int tl);
int recherche_candidat(Candidat **tc, int numc,int tl, int *trouve);
```

Ce sont deux algorithmes de recherche dichotomique coder d'une minière itérative (dans le struct.c).

Les algorithmes de tri :

ListeDept ajouter_departement(ListeDept ldept,char nom[31], int nbp, char resp[31]);

ListeDept ajouter_departement_en_tete(ListeDept ldept,char nom[31], int nbp, char resp[31]);

VilleIUT** ajouter_ville(VilleIUT *tiut[], int *tl, char nom[31], int pos);

Candidat** ajouter_candidat(Candidat *tc[], int *tl, int numc, float note[4], char nom[31], char prenom[31], int nbchoix,int pos);

ListeChoix ajouter_choix(ListeChoix lchoix,char ville[31], char dep[31], int desdp, int valcan);

ListeChoix ajouter_choix_en_tete(ListeChoix lchoix,char ville[31],char dep[31], int desdp, int valcan);

void ajouter_demande_ldept(ListeDept ldept,char dep[31], char nom[31], int numc, int desdp, int valcan);

demande* ajouter_demande(demande *dem, char nom[31], int numc, int desdp, int valcan);

demande* ajouter_demande_en_tete(demande *dem, char nom[31], int numc, int desdp, int valcan);

void ajouter_un_voeux(VilleIUT *tiut[], int tl, Candidat** tc, int tlc);

Je n'utilise aucun algorithme de tri car mes fonctions d'ajout sont des fonctions d'insertion qui, ainsi, tri les tableaux et liste chainé lors des insertions. Cela implique cert un chargement un peut plus long lors du démarrage mais il me permet d'éviter de devoir trié de nouveau à chaque modification des tableaux ou liste chainé.

La saisit contrôlée :

Je contrôle la saisit dans toutes les fonctions qui suivent. De plus, lorsque je ne contrôle pas la saisit (dans les fonctions du fichier function.c) je vérifie à l'aide des fonctions fonction de recherche (et les fonctions demande_existe ou departement_in_ville) si l'utilisateur utilise bien le programme comme prévu.

int menu_utilisateur(void);

int re_menu(void);

int sous_menu_utilisateur(void);

int sous_menu_admin(void);

int sous_menu_responsable(void);

Les fonctions récursives :

La majorité des fonctions d'affichage, d'insertion et de suppression sont coder de manière récursive (dans le struct.c). Cela entraine certes une plus grande utilisation de la mémoire, cependant, ça permet d'alléger le code et de le rendre plus lisible et compréhensible. Cependant, les fonctions de lecture/écriture dans les fichiers, de chargement ou de recherche sont codé de manière itérative affin de gagner en performance.

Les fonctions de menu :

Les menus prennent part dans la boucle while présent dans la fonction globale (dans le menu.c). A chaque tour de boucle, on vérifie si l'utilisateur est dans un sous menu (si typeUtilisateur != 0). Si oui, alors on affiche le sou menu qui correspond, si non, on affiche le menu principal. Une fois que l'on à choisit un sous menu et une fonction, la variable sous menu sera différente de 0 et ainsi pourra nous

permettre de choisir entre recommencer la même fonction ou revenir au sou menu où l'on se trouvait.

Les fonctions de lecture/écriture :

Les 11 fonctions de lectures et d'écritures ont un fonctionnement assez similaire (dans le main.c). Ces fonctions lissent ou écrivent dans des fichier texte (ce qui ait plus lent mais qui permet à un personnelle non formé d'interagir avec les fichiers en plus d'une meilleure portabilité). Ces fonctions sont codées de manière itérative pour les performances.