RAPPORT TP3 NF16

DOAN Nhat-Minh / Shuohui HU

OBJECTIFS

Il a pour objectif de se familiariser avec les listes chaînées et les différentes opérations nécessaires pour les manipuler, ici des données des étudiants.

SOURCES

- Fichier d'en-tête tp3.h, contenant la déclaration des structures/fonctions de base,

- Fichier source tp3.c, contenant la définition de chaque fonction,

- Fichier source main.c, contenant le programme principal.

PROGRAMMATION

1. Structures

```c
typedef struct Note
{

    float note;
    char *matiere;
    struct Note *suivant;

}T_Note;


typedef  T_Note *T_ListeNotes;
```

```c
typedef struct Etudiant
{

    int idEtu;
    char *nom;
    char *prenom;
    int nbreNote;
    T_ListeNotes Liste_Note;
    float moy;
    struct Etudiant *suivant;

}T_Etudiant;

typedef  T_Etudiant *T_ListeEtu;
```

On a déclaré deux structure Note et Etudiant avec leur types correspondants T_Note et T_Etudiant grâce à « typedef » puis deux type T_ListeNotes et T_Etudiant qui sont des pointeurs vers T_Note et T_Etudiant pour simplifier la gestion des listes chainées.

2.Fonctions

```
T_Note *creerNote(float note, char *matiere);

T_Etudiant *creerEtudiant(int idEtu, char *nom, char *prenom);

T_ListeNotes ajouterNote(float note, char *matiere, T_ListeNotes listeNotes);

T_ListeEtu ajouterNoteEtu(float note, char *matiere, int idEtu, T_ListeEtu listeEtu);

void afficheListeEtu(T_ListeEtu listeEtu);

T_ListeEtu supprimerNoteEtu(char *matiere, int idEtu, T_ListeEtu listeEtu);

T_ListeEtu ajouterEtu(int idEtu, char *nom, char *prenom, T_ListeEtu listeEtu);

T_ListeEtu supprimerEtu(int idEtu, T_ListeEtu listeEtu);

float calculMoyenne(int idEtu, T_ListeEtu listeEtu);

void afficherClassement(T_ListeEtu listeEtu);

void sousListes(char* matiere, T_ListeEtu listeEtu);
```

Ici on a choisi 3 fonctions supplémentaires. Ce sont ajouterEtu pour ajouter un étudiant dans la liste, calculMoyenne pour calculer la moyenne de chaque étudiant dans la liste et supprimerEtu pour supprimer un étudiant. Tous ces fonctions a le but de simplifier les calculs ainsi que les utiliser pour plusieurs fonctions différentes.

3.Complexités

n : nombre des étudiants

n_note : nombre des notes maximum d'un étudiant

- T_Note *creerNote(float note, char *matiere)

  T_Etudiant *creerEtudiant(int idEtu, char *nom, char *prenom)

=> Complexité O(1) car créer un note ou un étudiant chaque appel de la fonction.

- T_ListeNotes ajouterNote(float note, char *matiere, T_ListeNotes listeNotes)

=> Complexité O(1)

- T_ListeEtu ajouterNoteEtu(float note, char *matiere, int idEtu, T_ListeEtu listeEtu)

=> Complexité O(n) car on parcoure la liste et au pire cas où idEtu correspond au dernier étudiant.

- void affiche_ListeEtu(T_ListeEtu listeEtu)

=> Complexité O(n*n_note) car on parcoure la liste et au pire cas où chaque étudiant inscrit n_note notes.

- T_ListeEtu supprimerNoteEtu(char *matiere, int idEtu, T_ListeEtu listeEtu)

=> Complexité O(n+n_note) car on parcoure la liste, au pire cas où idEtu correspond au dernier étudiant et matière à supprimer correspond au dernier note.

- T_ListeEtu supprimerEtu(int idEtu, T_ListeEtu listeEtu)

=> Complexité O(n+n-1)=O(n) car on a utilisé 2 variables différents pour parcoure la liste et au pire cas où on fait une supprime en fin de la liste.

- T_ListeEtu ajouterEtu(int idEtu, char *nom, char *prenom, T_ListeEtu listeEtu)

=> Complexité O(n) car au pire cas où, l'idEtu que on veut saisir existe déjà, on doit parcoure jusqu'à la fin de la liste.

- void afficherClassement(T_ListeEtu listeEtu)

=> Complexité $O(n^2)$ car on appelle n fois la fonction afficherClassement dont la complexité O(n) au pire cas.

- float calculMoyenne(int idEtu, T_ListeEtu listeEtu)

=> Complexité O(n*n_note) car on parcoure la liste et pour chaque étudiant on parcoure jusqu'à la fin de la liste des notes, au pire cas où chaque étudiant inscrit à n_note notes.

- void sousListes(char* matiere, T_ListeEtu listeEtu)

=> Complexité $O(n^3*n\_note)$ car on parcoure la liste et pour chaque étudiant on parcoure jusqu'à la fin de la liste des notes, au pire cas où la matière se trouve à la fin de la liste, puis on considère le cas où tous les étudiant ont réussir l'UV (pareil pour rater et pas passer), chaque appel de listeEtuReussi =

ajouterEtu(tmp->idEtu, tmp->nom, tmp->prenom, listeEtuReussi); on a 1,2,3,..,n itérations dont la complexité $O(n^2)$.

CONCLUSION

On a travaillé à résoudre un problème ne pas compliqué d'un ensemble des données. Dans la vraie vie, il y aura beaucoup d'autres problèmes dans un système de gestion des données. La liste chainée est un outil pratique et fréquemment utilisé.