Compte-rendu du projet C de première année

La structure de nos fichiers :

Le fichier des logements :

```
typedef struct {
        int idLogement;
        char nomCite[20];
        char typeLogD[10];
        char dispo[4];
        char handicap[4];
        int idEtud;
}Logement;
```

Le fichier des étudiants :

```
typedef struct {
        int idEtud;
        char civilite[4];
        char nom[20];
        char prenom[20];
        char bourse[4];
        char handicap[4];
        int echellonBourse;
}Etudiant;
```

Le fichier des demandes :

```
typedef struct {
        int idDemande;
        int idEtudD;
        int echellonBourseD;
```

```
    char nomCiteD[20];
    char typeLogD[10];
}Demandes;
```

La structure des tableaux et Listes et leur mise en place :

Le tableau des logements : Tableau

```
Etudiant **tab;
```

NOUS AVONS CHOISI UN TABLEAU DE POINTEURS CAR :

- ➔ Plus optimisé qu'un tableau simple pour la mémoire de l'ordinateur mais moins qu'une liste.
- ➔ Cependant nous avons eu besoin de le trier par cité pour ensuite afficher et cela est plus simple sur un tableau de pointeurs qu'une liste.
- ➔ Également nous avons eu besoin de faire des modifications dessus cela est plus simple.

Ainsi c'est un équilibre entre performance et simplicité d'utilisation.

Le tableau des étudiants : Tableau

```
Logement  **tab;
```

NOUS AVONS CHOISI UN TABLEAU DE POINTEURS CAR :

- ➔ Même raisons que pour les étudiants et en plus la nécessité de faire une recherche dichotomique (plus rapide et moins couteux que le récursif des listes) et une insertion croissante par idEtudiant.

La liste des demandes : Liste

```
typedef struct listeDem{
    Demandes l;
    struct    listeDem *next;
}ListeDem, *Liste;
```

NOUS AVONS CHOISI UNE LISTE CAR :

- ➔ Nous n'avions pas le besoin trier la liste ni de rechercher dedans.
- ➔ L'insertion est simple.

➔ Et les listes sont la meilleure structure en terme d'optimisation pour la mémoire de l'ordinateur à notre connaissance.

LA LISTE :

Nom : LDem

Au début égale à NULL

Lecture dans un fichier binaire pour la remplir

Enregistrement dans un fichier binaire

Nos fonctions :

Manoah Levy-Valensi :

- **Logement** chargeLog(FILE *fic,int *nb, int *max);**

  Fonction permettant de charger un fichier de logements dans un tableau de pointeurs sur des type « Logement ».

  Opérationnel

- **Logement lireLog(FILE *fic);**

  Fonction qui lit un logement dans un fichier et le retourne à la fonction appelante.

  Opérationnel

- **void afficheLog(Logement l);**

  Fonction qui affiche le logement que la fonction appelante lui donne.

  Opérationnel

- **void afficheTabLog(Logement **tab, int nb);**

  Fonction qui affiche tous les logements à partir d'un tableau de pointeurs sur des logements.

  Il va se servir de la fonction « afficheLog » ci-dessus.

  Opérationnel

- **void afficheTabLogOccupe(Logement **tab,int nb);**

  Fonction qui affiche tous les logements occupés à partir d'un tableau de pointeurs sur des logements.

  Il va se servir de la fonction « afficheLog » ci-dessus.

  Opérationnel

- **void afficheTabLogLibre(Logement **tab,int nb);**

Fonction qui affiche tous les logements libres à partir d'un tableau de pointeurs sur des logements.

Il va se servir de la fonction « afficheLog » ci-dessus.

Opérationnel

- **void triFusionCiteTabLog(Logement **tab,int nb);**

  Fonction qui va regrouper par « noms de cités » par ordre alphabétique le tableau de pointeurs sur des logements passé par la fonction appelante.

  Il va faire appel aux fonctions « copie » et « fusion » ci-dessous.

  Opérationnel

  Amélioration possible : trier les regroupements de logements par ordre croissant de l'identifiant de logement.

- **void copie(Logement **tab,int i,int j, Logement **r);**

  Fonction qui va copier l'intégralité du premier tableau de pointeurs qu'elle reçoit dans le second. (Qui est 2x plus petit donc qui n'aura que la première ou deuxième moitié en fonction des arguments)

  Opérationnel

- **void fusion(Logement **r,int n,Logement **s,int m,Logement **tab);**

  Fonction qui assemble deux tableaux de taille logique "1" dans l'ordre alphabétique du nom de la Cité.

  Opérationnel

- **void sauvegardeLog(FILE *fic, Logement **tab, int nb);**

  Fonction qui va sauvegarder des logements dans le fichier d'origine.

  Opérationnel

- **int rechercheIdLog(int *idtemp,Logement **tab,int *nb);**

  Fonction qui demande à l'utilisateur le logement qu'il souhaite libérer, vérifie son existence et retourne sa position dans le tableau.

  Opérationnel

- **void libererLog(int *idtemp,Logement **tab,int u);**

  Fonction qui récupère le logement à libérer et le libère si nécessaire.

  Opérationnel

- **int menuLog(int *test);**

  Fonction qui affiche un menu textuel et retourne le choix de l'utilisateur.

  Opérationnel

Max Manin :

- **void test(void);**

  Fonction qui permet de créer la liste et qui vas appeler la fonction permettant de l'initialiser.

  Puis appelle la fonction menu en lui envoyant la liste initialisée.

  Opérationnel

- **Liste nouvelleListe(void);**

  Fonction qui retourne NULL pour initialiser la liste.

  Opérationnel

- **int reaffichermenu(void);**

  Fonction permettant de réafficher le menu principale selon le choix de l'utilisateur.

  Opérationnel

- **int menuDem(void);**

  Fonction permettant d'afficher un sous menu spécifique aux demandes pour ne pas surcharger le menu principale.

  Opérationnel

- **Liste insererDem(Liste LDem, Demandes d);**

  Fonction permettant d'insérer une demande dans la liste, soit lors de la lecture du fichier ou de l'ajout d'une demande grâce à la fonction insererEnTete.

  Opérationnel

- **int chargeFichierDem(Liste *LDem, char *fichier);**

  Fonction permettant grâce au nom du fichier donné d'ouvrir ce dernier et d'enregistrer son contenu dans une liste grâce à la fonction insererDem.

  Opérationnel

- **Liste insererEnTete(Liste LDem, Demandes d);**

  Fonction permettant d'incruster une demande au début de la liste passée.

  Opérationnel

- **Booleen listevide(Liste LDem);**

  Fonction permettant de savoir si la liste est vide ou non, elle revoie 1 ou 0 (vide / remplie).

  Opérationnel

- **Demandes lireDem(FILE *flot);**

  Fonction permettant de lire une demande dans le fichier binaire précédemment ouvert.

Opérationnel

- **void afficherliste(Liste LDem);**

  Fonction permettant d'afficher toutes les demandes mises dans la liste.

  Opérationnel

- **Liste choixSuppression(Liste LDem);**

  Fonction permettant de demander à l'utilisateur l'identifiant de la demande à supprimer et d'appeler la fonction supprimerDansListe.

  Opérationnel

- **Liste supprimerDansListe(Liste LDem, int s);**

  Fonction permettant de trouver la position de la demande dans la liste à supprimer puis appel la fonction supprimerEnTete.

  Opérationnel

- **Liste supprimerEntete(Liste LDem);**

  Fonction permettant de supprimer la demande quand l'on a trouvé sa position.

  Opérationnel

- **Liste insererDansListe(Liste LDem, Etudiant **tab, int *nb, int *max, char *fichier);**

  Fonction permettant de saisir les données pour rajouter une demande (id demande / id étudiant / bourse étudiant / nom citée / type d'appartement)

  Opérationnel

- **Booleen trouveID(Liste LDem, int d);**

  Fonction recherchant un identifiant de demande dans la liste et renvoie s'il y a l'identifiant ou von dans la liste (1 :oui / 0 :non) .

  Opérationnel

- **void LieuSauv(Liste LDem, Demandes d, int *sauv);**

  Fonction permettant de trouver où il faut enregistrer une demande pour facilitée l'enregistrement d'une seule demande dans un fichier.

  Opérationnel

  Mais il y a une amélioration possible c'est de fusionner cette fonction avec insererDem car l'on utilise ces deux fonction presque au même moment mais quand on les fusionnait il y avait des erreurs dans la liste (c'est pour cela que j'ai créé cette fonction).

- **void SauvElement(Demandes d, int *sauv, char *fichier) ;**

  Fonction permettant d'ouvrir le fichier binaire pour enregistrer une demande.

  Opérationnel

- **void sauvegardeElement(Demandes d, int *sauv, char * fichier, FILE *flot);**

Fonction permettant de sauvegarder une demande à un lieu précis en se déplaçant dans le fichier binaire.

Opérationnel

- **void sauvegarde(Liste LDem, char *fichier);**

  Fonction permettant d'ouvrir le fichier binaire pour pouvoir insérer la liste à l'intérieur.

  Opérationnel

- **void sauvegardeFich(Liste LDem, FILE *flot);**

  Fonction permettant d'insérer dans le fichier une demande.

  Opérationnel

- **void sauvFichTxt(Liste LDem, char *fichier);**

  Fonction permettant d'ouvrir ou de créer un fichier texte où l'on peut écrire les demandes pour qu'elle soit lisibles par quelqu'un quand l'on imprime le fichier.

  Opérationnel

- **void sauvegardeFichTxt(Liste LDem, FILE *flot);**

  Fonction permettant d'écrire les demandes dans le fichier texte.

  Opérationnel

Loup Rusak :

- Étudiant **chargeEtudiants(FILE *flot, int *nb, int *max) ;

  Fonction permettant de charger les étudiants du fichier des étudiants dans un tableau de pointeurs sur des étudiants.

  Opérationnel.

- Etudiant lireEtudiant(FILE *flot) ;

  Fonction permettant de lire un étudiant à partir du fichier des étudiants et de le renvoyer.

  Opérationnel. Possibilité d'améliorer en scannant des noms composés.

- void afficherEtudiant(Etudiant e) ;

  Fonction permettant d'afficher un étudiant à l'écran.

  Opérationnel. Possibilité d'améliorer en affichant de noms composés.

- void afficherTabEtud(Etudiant **tab, int nb) ;

  Fonction permettant d'afficher tous les étudiants du tableau de pointeurs sur des étudiants notamment en faisant appel à la fonction afficherEtudiant.

Opérationnel.

- int rechercherEtudiant(Etudiant **tab, int nb, Etudiant e) ;

  Fonction permettant de rechercher par idEtud un étudiant pour savoir s'il est déjà présent ou pas dans le tableau de pointeurs sur des étudiants.

  Opérationnel. Possibilité de l'améliorer en recherchant aussi le nom et le prénom pour pas insérer deux personnes de mêmes noms et prénoms.

- Etudiant **insertionEtudiant(Etudiant **tab, int *nb, int *max, Etudiant e) ;

  Fonction permettant d'insérer un étudiant donné dans le tableau de pointeurs sur des étudiants en fonction du résultat de la recherche de l'identifiant de l'étudiant.

  Opérationnel.

- void afficherMenuEtud(void) ;

  Fonction permettant d'afficher le texte et donc les options du menu des étudiants.

  Opérationnel.

- Etudiant saisieEtudiant(void) ;

  Fonction permettant de saisir toutes les informations sur un étudiant nécessaires à une future insertion ou recherche.

  Opérationnel.

- Etudiant** menuEtudiant(Etudiant **tab, int *nb, int *max) ;

  Menu général des étudiants. Fonction permettant de demander le choix de l'utilisateur et d'effectuer les actions en conséquence.

  Opérationnel.

- void sauvegardeEtudiant(FILE *flot, Etudiant **tab, int nb) ;

  Fonction permettant de sauvegarder dans le fichier de départ le tableau de pointeurs sur des étudiants après modifications.

  Opérationnel.

Vincent Vialette :

- **int rechercherEtudiantDem(Etudiant **tab, int nb, int e);**

  Fonction permettant de rechercher un étudiant et retourne la valeur de l'insertion.

  Opérationnel

- **int menuGestion ();**

  Sous Menu permettant d'attribuer un logement.

  Opérationnel

- **void gestionViaDem(Liste LDem, Etudiant **tabe, Logement **tabl ,int nbe, int nbl);**

Fonction permettant grâce à l'idDemande d'attribuer un logement.

Opérationnel, mais on pourrait faire afficher plusieurs logements qui correspondent à la demande de l'étudiant.

- **Demandes initialiseDem(Liste LDem, int s);**

  Permet d'initialiser une demande en fonction de l'idDemande

  Opérationnel

- **void gestionViaEtu(Liste LDem, Etudiant **tabe, Logement **tabl ,int nbe, int nbl);**

  Fonction permettant grâce à l'idEtudD d'attribuer un logement.

  Opérationnel, mais on pourrait faire afficher plusieurs logements qui correspondent à la demande de l'étudiant.

- **Demandes initialiseDem2(Liste LDem, int s);**

  Permet d'initialiser une demande en fonction de l'idEtudD

  Opérationnel

Max Manin & Vincent Vialette :

- **int menu(Liste LDem);**

  Fonction qui grâce à un entier qui représente le choix de l'utilisateur appelle une fonction qui répondra à l'attente de l'utilisateur.

  Opérationnel

- **int affichermenu(void);**

  Fonction permettant d'afficher les choix proposés et revoie à la fonction menu le choix de l'utilisateur.

  Opérationnel