Revu de projet

Le programme est divisé en 3 fichiers différents : main.c dans lequel on retrouve la fonction qui lance le fichier, projet.c qui contient les fonctions du programme, et projet.h dans lequel les fonctions sont définies avec les structures.

Tout d’abord nous avons défini les différentes structures dont nous aurions besoin par la suite dans le projet. Dans la structure emprunt nous avons mis la côte du livre dans une chaine de 10 caractères et la date d’emprunt dans une chaine de 11 caractères.

/\*Structure de Emprunt\*/

typedef struct{

    char cote[10];

    char date[11];

}Emprunt;

Ensuite nous avons défini la structure du lecteur avec le numéro de Lecteur (20 caractères), le nom (30 caractères), le prénom (30 caractères également), le code postal (renseigné par cp et contenant 6 caractères) et l’adresse qui est définie par la ville (30 caractères) et la rue(60 caractères)). Et enfin la liste d’emprunt qui contient 50 caractères.

/\*Structure de Lecteur\*/

typedef struct{

    char numLec[20];

    char nom[30];

    char prenom[30];

    char cp[6];

    char ville[30];

    char rue[60];

    Emprunt liste[50];

}Lecteur;

La structure d’ouvrage contient la côte qui est la même que dans la structure d’Emprunt, un titre de livre (30 caractères) et la catégorie du livre c’est-à-dire si c’est une BD, roman, etc (30 caractères).

/\*Structure d'Ouvrage\*/

typedef struct{

    char cote[30];

    char titre[30];

    char categorie[30];

}Ouvrage;

Et pour finir la structure du lecteur pour la liste chainée où on retrouve le numéro de lecteur le nom et le prénom avec les mêmes nombres de caractère.

/\*Structure de Lecteur\*/

typedef struct{

    char numLec[20];

    char nom[30];

    char prenom[30];

}LecTag;

Voici la structure de la liste chainée avec le lecteur ci-dessus.

/\*Structure de liste chainée pour Lecteur\*/

typedef struct liste{

    LecTag l;

    struct liste \*suivant;

}Maillon, \*ListeLecteur;

Nous avons débuté le programme par la création d’un tableau de pointeurs pour allouer dynamiquement le lecteur dans lequel nous avons enregistrer les adresses de Lecteur dans le fichier fe. Pour faire fonctionner cette partie nous avons d’abord lu le fichier fe ensuite nous l’avons chargé et enfin nous l’avons affiché.

Ensuite nous avons commencé à nous occuper de la partie ouvrages. Nous avons d’abord lu ouvrage dans le fichier fe. Ensuite nous avons chargé le fichier dans un tableau de pointeur avec l’allocation dynamique puis nous l’avons affiché.

Viens après le tri des lecteurs par ordre alphabétique. Pour cela nous avons utilisé une liste chainée avec les fonctions d’insertion Entête, insertion croissante, suppression et suppression croissante et enfin la fonction d’affichage.

Pour rechercher le lecteur par nom, nous somme parti du principe que chaque lecteur était unique dans la base de données, nous avons utilisé une recherche dichotomique pour le nom et le prénom. Nous avons ajouté une condition dans la fonction pour que la recherche fonctionne même si le positionnement du nom recherché se trouve en position 0.

Nous avons pensé à la possibilité d’ajouter un nouveau lecteur par le biais du clavier dans la base de données pour cela nous avons écrit une fonction qui lit l’entrée standard puis nous avons modifié le tableau pour ajouter le nouveau lecteur.

Après l’insertion nous avons procédé à la suppression du lecteur si jamais il se désinscrit de la bibliothèque. Pour cela nous avons une fonction qui permet la suppression et une fonction pour faire une mise à jour intermédiaire puis une mise à jour dans un fichier binaire.

Pour que les lecteurs puissent emprunter de nouveaux livres nous avons une fonction ajout Emprunt qui permet en demandant le nom du lecteur et en renseignant la référence de l’emprunt et la date l’ajout d’un nouvel emprunt au lecteur.

Nous avons ajouté une fonction au programme qui permet le débogage de la fonction qui lis un lecteur au clavier car à cause du « \n » il y avait un problème d’affichage. Cette fonction appelée clearBuffer permet de ne pas prendre en compte dans l’affichage les \n de fin de ligne.

Pour finir nous avons créé une fonction qui permet de d’afficher les différentes fonctions possibles sur le programme comme ajouter ou supprimer un lecteur et fait un menu permettant de choisir les fonctions que nous voulons effectuer sans avoir besoin de faire chaque étape du programme dans l’ordre. De plus nous avons ajouté des affichages pour expliquer ce qui vient d’être effectué au gérant de la bibliothèque.

La fonction appelante permet de prévenir l’utilisateur si jamais il y a un problème avec la fonction de chargement Lecteur.

Enfin voici la liste des fonctions incluent dans le programme.

/\*Fonctions Lecteurs\*/

Lecteur lireLec(FILE \*fe);

Lecteur lireLec2(Lecteur \*\*tLec, int \*n);

int chargementLecteur(char \*nomFich, Lecteur \*\*tLec, int tmax);

void affichageLec(Lecteur \*\*tLec, int n);

int rechDicoNom(Lecteur \*\*tLec, int n, char \*val);

int InscriptionLec(Lecteur \*\*tLec, int n, int tmax, Lecteur \*l);

void clearBuffer(void);

int rechDicoPrenom(Lecteur \*\*tLec, int n, char \*val);

int rechSequentiel(Lecteur \*\*tLec, int n, char \*val, int \*trouve);

int supprimeLec(Lecteur \*\*tLec, int n);

void ajoutEmprunt(Lecteur \*\*tLec, int n);

void miseajour(Lecteur \*\*tLec, int \*n);

ListeLecteur tabToList(Lecteur \*\*tLec, int n, ListeLecteur list);

void sauvegardeBin(Lecteur \*\*tLec, int n, Ouvrage \*\*o, int n2);

/\*Fonctions Ouvrages\*/

Ouvrage lire1Ouvrage(FILE \*fe) ; int chargementOuvrages(char \*nomFich, Ouvrage \*\*tOuv, int tmax);

void affichageOuvrage(Ouvrage \*\*tOuv, int n);

/\*Fonctions Lecteur ordre alphabétique\*/

ListeLecteur listeVide(void);

ListeLecteur insertionEnTete(ListeLecteur list, LecTag l);

ListeLecteur insertionCroissante(ListeLecteur list, LecTag l);

ListeLecteur supprimeListe(ListeLecteur list);

ListeLecteur suppressionCroissante(ListeLecteur list, LecTag l); void afficherEnsemble(ListeLecteur list);

/\*Fonctions Menu\*/

void selectGui(int \*val);

void menu(Lecteur \*\*tLec, int n);

/\*Fonction Test\*/

void test(void);