

Explications

Projet Bibliothèque

I/ Table des matières .c

1. LECTURE données fichier

1.1 Lecture de structure	15
1.1.1 Lecteur lireLecteur(FILE *f);	17
1.1.2 Ouvrage lireOuvrage(FILE *f);	43
1.1.3 Date lireDate(FILE *f);	69
1.1.4 Emprunt lireEmprunt(FILE *f);	77
1.2 Affichage structure	97
1.2.1 void reduireLongueur(char *nom, char *nouveauNom, int taille);	99
1.2.2 void ajouterBlanc(char *nom, int taille);	119
1.2.3 void afficherLecteur(Lecteur l);	132
1.2.4 void afficherCarteLecteur(Lecteur l);	155
1.2.5 void afficherOuvrage(Ouvrage o);	166
1.2.6 void afficherCarteOuvrage(Ouvrage o)	190
1.2.7 void afficherDate(Date d);	204
1.2.8 void afficherEmprunt(Emprunt e);	209
1.3 Affichage Tableau/Liste	222
1.3.1 void afficherTableauLecteur(Lecteur **tab, int nb);	223
1.3.2 void afficherTableauOuvrage(Ouvrage **tab, int nb);	238
1.3.3 void afficherListeEmprunt(ListeEmprunt l);	251

2. CHARGEMENT dans tableau / liste

2.1 Lecteur **chargementLecteur(char *nomFichier, int *taille, int *tmax);	272
2.2 Ouvrage **chargementTableauOuvrage(char *nomFichier, int *tmax);	327
2.3 ListeEmprunt listeVide(void);	366
2.4 ListeEmprunt chargementListeEmprunt(char *nomFichier);	371

3. INSERTION d'un élément

3.1 Création du lecteur, ouvrage, emprunt	405
3.1.1 Bool verificationNumTel(char *numTel);	406
3.1.2 Lecteur creerLecteur(int numLect);	433
3.1.3 Ouvrage creerOuvrage(Ouvrage **tab, int nb);	472
3.1.4 Emprunt creerEmprunt(Lecteur **tab, int nb, Ouvrage **tabOuv, int nbOuv, int *codeRetour);	521
3.2 tri par fusion (tableau lecteur)	580
3.2.1 void copierLecteur(Lecteur **R, int i, int j, Lecteur **tab);	581
3.2.2 void fusionLecteur(Lecteur **R, int n, Lecteur **S, int m, Lecteur **tab);	593
3.2.3 int comparerNomPrenom(char *nom1, char *prenom1, char *nom2, char *prenom2);	645
3.2.4 void triDicoLecteur(Lecteur **tab, int nb);	655

3.3 tri par échange (tableau ouvrage)	685
3.3.1 void echangerOuvrage(Ouvrage **tab, int i, int j);	686
3.3.2 int plusGrand(Ouvrage **tab, int nb);	696
3.3.3 void triEchangeOuvrage(Ouvrage **tab, int nb);	706
3.4 recherche dichotomique	719
3.4.1 int rechercheDicoLecteur(Lecteur **tab, int numLect, char *nom, char *prenom, int nb, int *trouve);.....	720
3.4.2 int rechercheDicoOuvrage(Ouvrage **tab, char *cote, int nb, int *trouve);	757
3.5 insertion.....	781
3.5.1 void decalageADroiteLecteur(Lecteur **tab, int pos, int nb);	782
3.5.2 void insertionLecteur(Lecteur l, Lecteur **tab, int *nb, int *tmax);	791
3.5.3 void decalageADroiteOuvrage(Ouvrage **tab, int pos, int nb);	843
3.5.4 void insertionOuvrage(Ouvrage o, Ouvrage **tab, int *nb);	850
3.5.5 ListeEmprunt insererEnTete(ListeEmprunt l, Emprunt e);	887
3.5.6 int comparerEmprunt(Emprunt e1,Emprunt e2);	904
3.5.7 ListeEmprunt inserer(ListeEmprunt l, Emprunt e);	912
 4. SUPPRESSION d'un élément	
4.1 void decalageAGaucheLecteur(Lecteur **tab, int pos, int nb);	939
4.2 void supprimerLecteur(Lecteur **tab, int numLect, char *nom, char *prenom, int *nb);.....	947
4.3 void decalageAGaucheOuvrage(Ouvrage **tab, int pos, int nb);	973
4.4 void supprimerOuvrage(Ouvrage **tab, char *cote, int *nb);	979
4.5 ListeEmprunt supprimerEnTete(ListeEmprunt l);	1005
4.6 int comparerDate(Date d1, Date d2);	1021
4.7 ListeEmprunt supprimer(ListeEmprunt l, Emprunt e, int *codeRetour);	1031
 5. MODIFICATION d'un élément	
5.1 void modifierLecteur(Lecteur **tab, int numLect, char *nom, char *prenom, int nb);	1070
5.2void modifierOuvrage(Ouvrage **tab, char *cote, int nb);	1093
5.3 ListeEmprunt modifierEmprunt(ListeEmprunt l, Emprunt e);	1117
 6. SAUVEGARDE dans un fichier	
6.1 Fichier texte	1157
6.1.1 void sauvegarderTableauLecteur(char *nomFichier, Lecteur **tab, int nb);	1158
6.1.2 void sauvegarderTableauOuvrage(char *nomFichier, Ouvrage **tab, int nb);	1185
6.2 Fichier binaire	1217
6.2.1 void sauvegarderListeEmprunt(char *nomFichier, ListeEmprunt l);	1218
6.3 Globale	1247
6.3.1 void sauvegarder(char *nomFichierLecteur, Lecteur **tabLect, int nbLect, char *nomFichierOuvrage, Ouvrage **tabOuv, int nbOuv, char *nomFichierEmprunt, ListeEmprunt l);.....	1248
 7. MENU	
7.1 void afficherMenu(void);	1263
7.2 void afficherSousMenu(void);	1291
7.3 int choixMenu(int nbPossibilite);	1306

8. FONCTIONS de recherche diverses

8.1 void rechercher(Lecteur **tab, int nb, Ouvrage **tabOuv, int nbOuv, ListeEmprunt l);1327

9. GLOBAL

9.1 void global(void)1470

II/ Modèle entité-association



III/ Affichage

Menu :

```
C:\Users\Clément\Documents\C_Program\project\bin\Debug\project.exe
Menus :
Fonctions Lecteurs:
1-Afficher tous les lecteurs
2-Insérer un nouveau lecteur
3-Supprimer un lecteur
4-Modifier les informations d'un lecteur
Fonctions Ouvrages:
5-Afficher tous les ouvrages
6-Insérer un nouvel ouvrage
7-Supprimer un ouvrage
8-Modifier les informations d'un ouvrage
Fonctions emprunts:
9-Afficher tous les emprunts
10-Insérer un nouvel emprunt
11-Supprimer un emprunt
12-Modifier un emprunt
13-Recherches diverses
14-Sauvegarder et quitter
Entrez un numéro d'action à effectuer <-1 pour quitter> :
```

Sous-Menu :

```
C:\Users\Clément\Documents\C_Program\project\bin\Debug\project.exe
Fonctions de recherche :
1-Rechercher les ouvrages empruntés par un lecteur
2-Rechercher la liste des ouvrages non rendu à la date limite
3-Rechercher les lecteurs d'une ville
4-Rechercher les ouvrages d'une certaine catégorie
Fonctions d'affichage :
5-Afficher un lecteur
6-Afficher un ouvrage
7-Afficher un emprunt
Entrez un numéro d'action à effectuer <-1 pour quitter> : _
```

Tableaux :

Lecteurs :

Num	Non	Prenom	Ville	NumTel	Retard	Emprunt
4	BECOUZE	Florent	Magnet	0980832343	9	2
1	DUPONT	Albert	Clermont-Ferr..	0778359213	0	0
5	DUPONT	Albert	Clermont-Ferr..	0778359213	0	0
6	MANN	Etienne	St-Germain	0625773194	0	0
2	MARTIN	Roger	Vichy	0625774574	4	1
3	TORTI	Clement	St-Germain	0615847596	3	1

Souhaitez-vous continuer ? <0:N> : _

Ouvrages :

Cote	Titre	Categorie	Auteur	Emprunte (Oui/Non)
B10B3	Tintin au Tib..	BD	Herge	OUI
F20C4	Harry Potter ..	Fantaisi..	JK Rowli..	OUI
S50F6	Soul Eater	Manga	Okubo	OUI
S52F1	Sciences & Vi..	Science	*****	OUI
T123B	Lapin cretin	BD	???	NON
Souhaitez-vous continuer ? <0:N> : _				

Emprunts :

Num	Cote	Date Emprunt	Date Retour
2	F20C4	14/09/17	14/10/17
3	S52F1	02/05/17	02/06/17
4	B10B3	27/12/17	30/01/18
4	S50F6	26/03/17	26/04/17
Souhaitez-vous continuer ? <0:N> : _			

Fichiers :

Lecteurs :

```
5
DUPONT
Albert
Clermont-Ferrand
0778359213
0
0
6
MANN
Etienne
St-Germain
0625773194
0
0
```

Ouvrages :

```
5
B10B3
Tintin au Tibet
BD
Herge
0
F20C4
Harry Potter 4
Fantaisie
JK Rowling
0
S50F6
Soul Eater
Manga
Okubo
0
```

Emprunts :

```
STXNULNULNULF20C4NULSONULNULNUL
NULNULNULDC1NULNULNULSONULNULNUL
NULNULNULDC1NULNULNULETXNULNULNULS52F1NULSTXNULNULNULENQNULNUL
NULDC1NULNULNULSTXNULNULNULACKNULNULNULDC1NULNULNULEOTNULNULNUL
B10B3NULESCNULNULNULFBNULNULNULDC1NULNULNULRSNULNULNULSOHNULNUL
NULDC2NULNULNULEOTNULNULNULS50F6NULSUBNULNULNULETXNULNULNULDC1
NULNULNULSUBNULNULNULEOTNULNULNULDC1NULNULNUL
```

IV/ Stratégie

- Lecteurs :

- Les lecteurs sont enregistrés dans un fichier texte et chargés dans un tableau dynamique de pointeur vers des structures Lecteur.
- On ne connaît pas le nombre de lecteur à l'ouverture du fichier.
- Les lecteurs sont triés selon leur nom et prénom (éventuellement par numéro lecteur) par ordre alphabétique. On peut avoir plusieurs lecteurs ayant le même nom et prénom, leur identifiant (numLecteur) sera différent.
- Le numéro lecteur est un entier positif unique, fixe et définie automatiquement à la création d'un lecteur comme étant égale au numeroLecteur le plus grand avant l'insertion + 1.
- Le tri se fait par fusion et la recherche est dichotomique.
- Le champ « nbEmprunt » correspond au nombre d'occurrence du lecteur dans la liste d'emprunt et est automatiquement mis à jour.
- Un lecteur ayant plus de 10 retards se voit attribuer une amende de 5 euros (amende non matérialisée, seulement énoncée) et son nombre de retard retourne à 0.
- La suppression d'un lecteur est possible si et seulement si il n'a pas d'emprunt en cours.

- Ouvrages :

- Les ouvrages sont enregistrés dans un fichier texte et chargés dans un tableau dynamique de pointeur vers des structures Ouvrage. La différence est que le nombre d'ouvrage est donné en début de fichier. Taille logique et physique sont alors les mêmes.
- On trie les ouvrages par ordre alphabétique de leur cote qui est propre à chaque ouvrage (même des ouvrages de même titre).
- Le tri se fait par échange et la recherche est dichotomique.
- Son champ « emprunte » varie automatiquement en fonction de s'il se trouve ou non dans la liste d'emprunt.
- La suppression d'un ouvrage est possible si et seulement si il n'est pas en cours d'emprunt.
- La catégorie d'un ouvrage appartient à un ensemble fini, défini au préalable (BD, Fantaisie ...).

- Emprunt :

- Les emprunts sont lus et enregistrés dans un fichier binaire.
- On ne connaît pas le nombre d'emprunt et ceux-ci sont chargés dans une liste chaînée composée d'un maillon (emprunt) et d'une adresse vers l'élément suivant. Pas de pointeur.
- Le tri se fait par le numéro de lecteur de l'emprunteur, puis par cote.
- Le tri et la recherche sont séquentiels.
- La date de retour est par défaut égale au mois suivant l'emprunt. Modifiable.
- Un emprunt peut être effectué si et seulement si le lecteur et l'ouvrage existe, si le lecteur a moins de 10 emprunts en cours et si l'ouvrage n'est pas déjà emprunté.
- Un éventuel retard est pris en compte, une fois l'ouvrage rendu.

- MENU :

- Un menu à choix multiple apparaît dès le lancement du programme.
- Un choix est demandé, vérifié et le « case » correspondant du switch est effectué.
- A chaque action effectuée, une demande de continuer apparaît. Si l'utilisateur répond « N », le programme se termine et les tableaux et la liste sont sauvegardés dans les fichiers d'origines.
- Cas particulier : « case 13 ». Un nouveau menu avec des fonctionnalités de recherche supplémentaires s'ouvre.

V/Explication

Pour des raisons de longueur, seules les fonctions ambiguës seront détaillées (les sources d'ambiguïté sont soulignées).

1. void reduireLongueur(char *nom, char *nouvNom, int taille);

void ajouterBlanc(char *nom, int taille) ;

Problème à résoudre : Certains champs peuvent être très longs (ville, nom ouvrage...) ou trop courts et s'affichent mal dans les tableaux.

But : Ces deux fonctions permettent un alignement des éléments des tableaux et liste affichés.

Fonctionnement :

(Pour les occurrences trop longues) On crée, pour chacun des champs à afficher, une nouvelle chaîne de caractère *nouvNom*, qui est une copie éventuellement plus courte de *nom*.

Cette chaîne de caractères aura au maximum *taille* caractères. (PS : si *nom* a plus de *taille-2* caractères, deux petits points sont ajoutés à la fin de *nouvNom*).

(Pour les occurrences trop courtes) Ajouter autant de blanc à la fin de chaque champs qu'il faut pour que *nom* ait *taille* caractères.

2. Bool verificationNumTel(char *numTel);

Problème à résoudre : On souhaite éviter les fautes de frappe lors de la saisie d'un numéro de téléphone.

But : Tous les numéros de téléphone doivent avoir 10 chiffres et pas de lettre.

Fonctionnement : On parcourt *numTel* et on compte le nombre de chiffres. Si ce nombre est différent de 10 ou si un des caractères de *numTel* n'est pas un chiffre, on retourne **false**.

3. int comparerNomPrenom(char *nom1, char *prenom1, char *nom2, char *prenom2);

int comparerEmprunt(Emprunt e1, Emprunt e2);

int comparerDate(Date d1, Date d2);

Problème à résoudre : On souhaite comparer des structures dans des conditionnels.

But : Optimiser les recherches et les tris ou vérifier la validité d'un champ.

Fonctionnement : Similaire au « strcmp() », ces fonctions retournent un nombre négatif si (respectivement).

- *nom1* et *prenom1* sont situés avant *nom2* et *prenom2* par ordre alphabétique.

- *e1.numLecteur* < *e2.numLecteur* ou (*e1.numLecteur* = *e2.numLecteur* et *e1.cote* avant *e2.cote*).

- *d1* est antérieur à *d2*.

4. void sauvegarder(char *nomFichierLecteur, Lecteur **tabLect, int nbLect, char *nomFichierOuvrage, Ouvrage **tabOuv, int nbOuv, char *nomFichierEmprunt, ListeEmprunt l);

Fonctionnement : Se charge d'appeler les différentes fonctions de sauvegarde, afin de limiter la redondance au niveau du code.

5. int choixMenu(int nbPossibilite);

Problème à résoudre : éviter la redondance de code en écrivant plusieurs fois les mêmes fonctions.

But : Avoir une fonction qui peut correspondre à plusieurs menus ayant un nombre de choix différents.

Fonctionnement : Tant que le nombre saisi par l'utilisateur n'est pas compris entre 1 et *nbPossibilite* ou bien n'est pas égale à -1, l'utilisateur doit retaper.

6. void rechercher(Lecteur **tab, int nb, Ouvrage **tabOuv, int nbOuv, ListeEmprunt l);

Problème à résoudre : Plus de place à l'écran pour proposer de nouvelles fonctions sans devoir « scroller ».

But : Créer un sous-menu avec des fonctionnalités de recherche afin d'améliorer la visibilité à l'utilisateur et réduire le nombre de ligne de code dans la fonction global.

Fonctionnement : Si l'utilisateur choisi l'option 13, un sous-menu apparait. Le fonctionnement est le même que pour le Menu, mise à part qu'une seule fonction est exécutée avant de retourner au menu principal.

Les paramètres sont essentiels pour les différentes recherches proposées.