

Rapport Structure de donnée TP 1

Encadrant : Daniel Yves Jean

Mars 2020

Sommaire

Description du TP.....	3
Structures des données	3
Format des fichiers textes.....	5
Arborescence du projet.....	7
Makefile.....	8
Mode d'emploi	9
Fonctions et Procédures	11
Lists.h	11
Borrow.h	12
Menu.h.....	16
Traces.....	17
Jeux d'essais, liste des cas.....	26
Cas General	26
Annexes	33
makefile	33
structures	34
main	34
List.....	42
Borrow.....	47

Description du TP

L'objectif du TP est de concevoir une bibliothèque dont les livres, regroupés par catégories, peuvent être empruntés et rendus.

Structures des données

Les données sont stockées en mémoire dans deux listes chainées différentes. La première liste chainée appelée « library » contient les livres rangés dans leur catégorie. La seconde appelée « borrowings » contient les livres qui ont été empruntés.

Library :

Library est la liste chainée des livres regroupés par catégories. Les maillons qui la composent sont des structures nommées « library_t ». Ils contiennent le nom de la catégorie, un pointeur vers le premier livre d'une sous liste chainée appelée « books » qui contient les livres de cette catégorie, et un pointeur vers l'élément suivant.

Les maillons qui composent la sous liste chainée « books » sont des structures nommées « books_t ». Ils contiennent les détails des livres. Un numéro unique, le titre du livre, son statut (disponible ou emprunté) ainsi qu'un pointeur vers le l'élément suivant.

Cette sous liste « books » est triée par ordre croissant des numéros des livres.

library_t		
begBooks	category	next
NULL	BD	NULL

Schéma structure library_t

books_t			
next	title	bookNb	isBorrowed
NULL	tintin	21	0

Schéma structure books_t

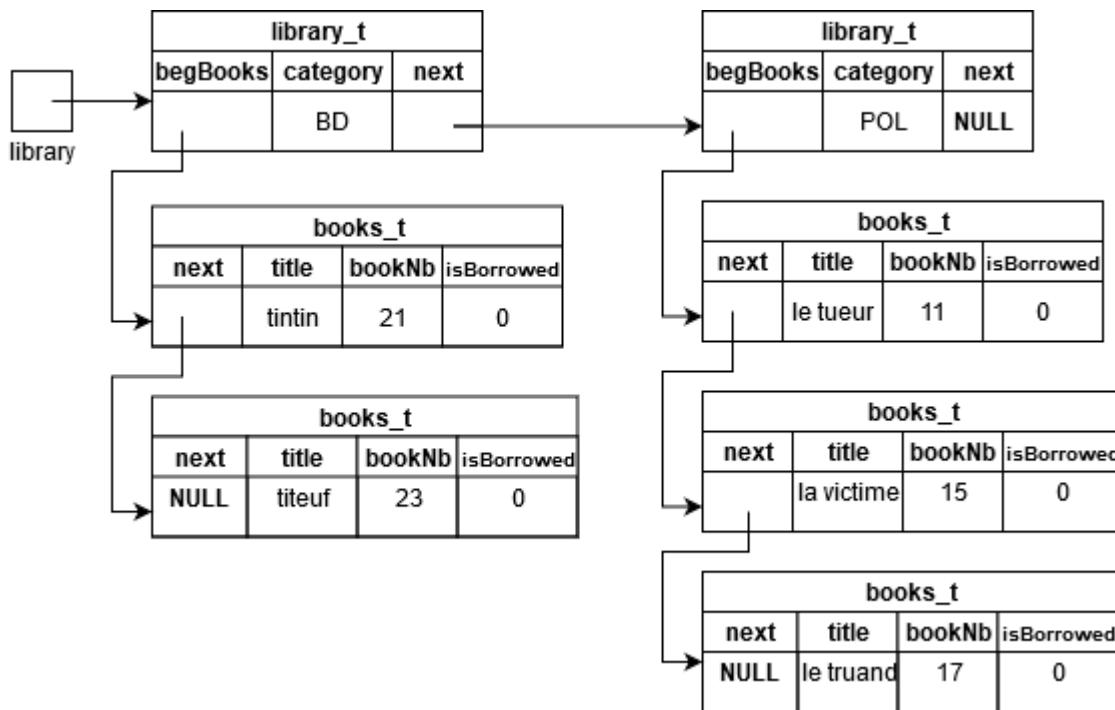


Schéma d'un exemple de structure pour library

Amélioration possible de la structure imposée : ajouter un pointeur sur le dernier livre d'une catégorie pour éviter de parcourir tous les livres de la catégorie si l'on souhaite implémenter une fonctionnalité permettant d'ajouter un livre pour agrandir la bibliothèque (dans l'idée d'un programme plus modulaire)

Borrowings :

Borrowings est la liste chainée des livres empruntés. Les maillons qui la composent sont des structures nommées « borrowings_t ».

Chaque maillon contient le numéro du livre emprunté et sa date de retour.

borrowings_t		
bookNb	returnDate	next
21	20200112	NULL

Schéma structure borrowings_t

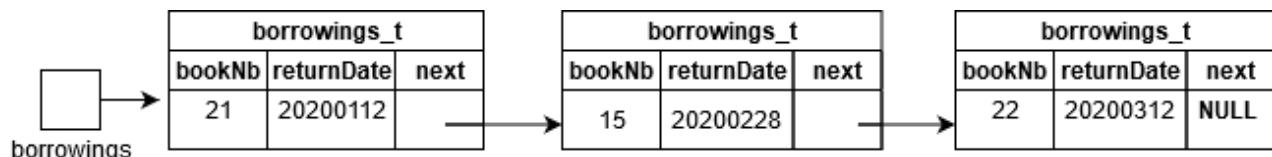


Schéma d'un exemple de structure pour borrowings

Amélioration possible de la structure imposée : ajouter le nom de la catégorie auquel appartient le livre. Pour éviter de parcourir la liste library lors de la sauvegarde de la liste des emprunts.

Format des fichiers textes

Fichiers des livres de la bibliothèque : library.txt

La liste des livres est enregistrée dans un fichier pour qu'à chaque utilisation du programme, tous les livres soient automatiquement ajoutés à la bibliothèque.

Structure :

<nom de la catégorie> <nombre de livre dans la catégorie>

<numéro du livre> <titre du livre>

...

...

<nom de la catégorie> <nombre de livre dans la catégorie>

<numéro du livre> <titre du livre>

...

...

Exemple :

```
POL 3
11 le tueur
15 la victime
17 le truand
BD 2
21 tintin
23 titeuf
JAP 0
CAL 1
28 caligros
```

Fichier des emprunts : borrowings.txt

La liste des livres empruntés est enregistrée dans un fichier pour sauvegarder les livres qui sont empruntés et leur date de retour.

Structure :

<nom de la catégorie du livre> <numéro du livre> <date de retour>

...

....

Exemple :

```
POL 15 20191112
BD 21 20200122
GHE 64 45213698
CAL 28 20180123
```

Fichier des livres à rendre : brought_back.txt

Tous les livres rendus sont enregistrés dans un fichier permettant d'actualiser l'état des livres rendus et modifier la liste des livres empruntés.

Structure :

<catégorie du livre> <numéro du livre>

Exemple :

BD 21
CAL 28

Arborescence du projet

```
bin
└── executable
borrowings
├── 2020-02-18_18h32'01'
├── 2020-02-20_19h22'31'
└── 2020-02-25_16h08'49'
build
├── borrow.d
├── borrow.o
├── lists.d
├── lists.o
├── main.d
├── main.o
├── menu.d
└── menu.o
makefile
src
├── borrow.c
├── borrow.h
├── common.h
├── lists.c
├── lists.h
├── main.c
├── menu.c
└── menu.h
text_files
├── README.txt
├── borrowings.txt
└── brought_back.txt
library.txt
```

Makefile

Un Makefile est un fichier constitué de plusieurs règles de la forme :

cible : dépendances
 commandes

Cibles :

- all** : regroupe dans ses dépendances l'executable à produire.
- clean** : permet de supprimer tous les fichiers intermédiaires.
- .PHONY** : permet de reconstruire les dépendances de la cible (dépendante de .PHONY), dans le cas où des fichiers porteraient le même nom.

Variabes :

- CFLAGS** : regroupe les options de compilation.
 - Wall -Wextra : warnings de compilation
 - g : Génère les informations de débogage.
 - MMD : permet la génération des fichiers .d pour les dépendances.

- LIB** : regroupe les options de l'édition de liens.
 - lm : bibliothèque maths

- SRC** : contient la liste des fichiers sources du projet. La commande wildcard permet l'utilisation du joker * dans la définition d'une variable.

- OBJ** : contient la liste des fichiers objets. OBJ est rempli à partir de SRC.
 - \$**(patsubst pattern, replacement, text) : Trouve les mots dans text qui correspondent au pattern et les remplace par replacement.

- DEP** : fichiers .d contenant les dépendances associées à un fichier .o du même nom.

Variabes internes :

- \$@** : Le nom de la cible.
- \$<** : Le nom de la première dépendance.
- \$^** : La liste des dépendances.
- @** : Permet de ne pas afficher la commande dans la console.

Mode d'emploi

Le programme fonctionne de la manière suivante :

Exécuter le programme : bin/executable text_files/library.txt

Une fois l'exécutable lancé, le programme effectue une vérification du fichier Livres qui va être lu.

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
 Oui  :1
 Non  :0
 -: |
```

Entrer 1 pour lire le fichier, 0 pour quitter.

Un message indique à partir de quels fichiers les listes ont été créées.

```
Création de la bibliothèque depuis : text_files/library.txt
Actualisation des livres empruntés depuis : text_files/borrowings.txt
Actualisation des livres rendus depuis : text_files/brought_back.txt
```

```
MENU
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter                      : 0
-: |
```

Vient ensuite le menu, où l'utilisateur peut effectuer des actions.

Voici les options que l'utilisateur peut effectuer :

- 1 : Affiche la bibliothèque
- 2 : Affiche la liste des emprunts
- 3 : Lit le fichier des retours
- 4 : Affiche la liste des emprunts à rendre avant le...
- 5 : Sauvegarde les emprunts dans un fichier
- 0 : Quitter le programme

Option 3 : Lit le fichier des retours :

```
Nom du fichier des rendus :
text_files/brought_back.txt  : 1
Autre nom à entrer          : 0
-: 0

Entrer le nom du fichier des retours : |
-: |
```

L'utilisateur à la possibilité de lire automatiquement le fichier text_files/brought_back.txt en sélectionnant 1.

S'il choisit 0, il peut alors entrer le nom d'un autre fichier qui sera lu pour actualiser la liste des retours.

Option 4 : Affiche la liste des emprunts à rendre avant le... :

```
| Selectionner une date AAAAMMMJJ : |  
-: |
```

L'utilisateur doit alors entrer une date de la forme AAAAMMMJJ pour afficher les listes des emprunts à rendre avant cette date.

Option 5 : Sauvegarde les emprunts dans un fichier :

```
fichier créé : borrowings/2020-03-07_19h10'49''
```

Un nouveau fichier des emprunts est alors créé, le programme indique son nom.
Son contenu est aussi affiché.

C'est le fichier créé avec cette option dernier en date qui sera lu automatiquement lors de la prochaine exécution du programme pour actualiser la liste des emprunts.

Option 0 : Quitter le programme :

```
On quitte
```

Quitte simplement le programme.

Messages d'erreurs :

Un exemple de certains messages d'erreurs que l'utilisateur peut être amené à lire si ce dernier entre des valeurs incorrectes dans les menus. Ces derniers sont écrits en rouge.

```
Création de la bibliothèque depuis : unfichierinexistant  
Bibliothèque vide  
  
Nom de fichier pour les retours inexistant  
  
Choix Invalide  
  
Selectionner une option  
  
Date Invalide
```

Fonctions et Procédures

Lists.h

displayLibrary :

Affiche la bibliothèque.

Paramètres :

Nom	Type	Détail
curLib	library_t const *	Pointeur sur la bibliothèque (par adresse)

Valeur de retour : void

createCategory :

Créer une catégorie et l'ajoute dans la bibliothèque (en tête).

Paramètres :

Nom	Type	Détail
library	library **	Pointeur sur la bibliothèque (par adresse)
category	char [4]	Nom de la catégorie en 3 caractères maximum

Valeur de retour : Retour d'erreur 1 si réussi, 0 sinon.

createBook :

Créer un livre et l'ajoute à la fin de la liste des livres.

Paramètres :

Nom	Type	Détail
library	library_t **	Pointeur sur la bibliothèque (par adresse)
lastBook	book_t *	Pointeur sur le dernier livre de la catégorie (par adresse)
bookNb	int	Numéro du livre
title	char [11]	Nom du livre en 10 caractères maximum

Valeur de retour : Retour d'erreur 1 si réussi, 0 sinon.

createLibrary :

Créer la bibliothèque.

Paramètres :

Nom	Type	Détail
filename	char *	Nom du fichier à lire pour remplir la bibliothèque
library	library_t **	Pointeur sur la bibliothèque (par adresse)

Valeur de retour : Retour d'erreur 1 si réussi, 0 sinon.

freeLibrary :

Libère la mémoire la bibliothèque.

Paramètres :

Nom	Type	Détail
curLib	library_t const *	Pointeur sur la bibliothèque (par adresse)

Valeur de retour : void

remove_endstr_r_windows :

Supprime le caractère '\r ' de fin de chaîne si il existe.

Paramètres :

Nom	Type	Détail
line	char *	Chaine de caractère

Valeur de retour : void

Borrow.h

displayBorrowings :

Affiche la liste des emprunts.

Paramètres :

Nom	Type	Détail
curBorrow	borrowings_t const *	Pointeur courant sur la liste des emprunts

Valeur de retour : void

borrowBook :

Lit le fichier des emprunts et rempli la liste chaînée.

Paramètres :

Nom	Type	Détail
filename	char *	Nom du fichier des emprunts
library	library_t	Pointeur sur la bibliothèque
borrowings	borrowings_t	Pointeur sur la liste des emprunts

Valeur de retour : void

isBookInLibrary :

Cherche dans la bibliothèque si le livre que l'on veut emprunter existe.

Paramètres :

Nom	Type	Détail
curLib	library_t const *	Pointeur sur la bibliothèque
category	char [4]	Nom de la catégorie du livre recherché
bookNb	int	Numéro du livre recherché

Valeur de retour : pointeur contenant l'adresse du livre de la bibliothèque que l'on souhaite emprunter, NULL si il n'est pas trouvé.

insertBorrowing :

Insère le livre emprunté dans la liste des emprunts et modifie la valeur de isBorrowed du livre dans la bibliothèque.

Paramètres :

Nom	Type	Détail
borrowings	library_t const *	Pointeur sur la liste des emprunts (par adresse)
bookBorrowed	books_t *	Adresse du livre dans la bibliothèque que l'on souhaite emprunter
Date	char [9]	Date de retour du livre (de la forme AAAAMMJ)

Valeur de retour : void

broughtBackBook :

Lit le fichier des retours, supprime les livres rendus de la liste des emprunts.

Paramètres :

Nom	Type	Détail
filename	char *	Nom du fichier des retours
library	library_t *	Pointeur sur la bibliothèque
borrowings	borrowings_t **	Pointeur sur la liste des emprunts (par adresse)

Valeur de retour : void

deleteBorrowing :

Supprime de la liste des emprunts le livre dont le numéro est passé en paramètre.

Paramètres :

Nom	Type	Détail
borrowings	borrowings_t **	Pointeur sur la liste des emprunts (par adresse)
bookNb	int	Numéro du livre à supprimer

Valeur de retour : void

isBorrowedToFalse :

Passe la valeur de isBorrowed de la bibliothèque à False.

Paramètres :

Nom	Type	Détail
curLib	library_t *	Pointeur sur la bibliothèque
category	char [4]	Nom de la catégorie du livre
bookNb	int	Numéro du livre

Valeur de retour : void

displayBorrowingsBeforeDate :

Affiche le numéro et la date de retour des livres à rendre avant une date.

Paramètres :

Nom	Type	Détail
curBorrow	borrowings_t const *	Pointeur sur la liste des emprunts
date	char [9]	Date

Valeur de retour : void

createFilename :

Remplie une chaîne de caractère avec la date et l'heure actuel (AAAA-MM-JJ_hhHm'm'ss").

Paramètres :

Nom	Type	Détail
filename	char *	Chaine de caractère à remplir

Valeur de retour : void

saveBorrowingsInFile :

Sauvegarde les emprunts dans un fichier.

Paramètres :

Nom	Type	Détail
filename	char *	Nom du fichier
library	library_t const *	Pointeur sur la bibliothèque
curBorrow	borrowings_t const *	Pointeur sur la liste des emprunts

Valeur de retour : void

findCategoryName :

Remplie une chaine de caractère avec le nom de la catégorie correspondant au numéro du livre.

Paramètres :

Nom	Type	Détail
curLib	library_t const *	Pointeur sur la bibliothèque
bookNb	int	Numéro du livre
category	char [4]	Chaine de caractère à remplir

Valeur de retour : void

freeBorrowings :

Libère la mémoire de la liste des emprunts.

Paramètres :

Nom	Type	Détail
borrowings	borrowings_t **	Pointeur sur la liste des emprunts (par adresse)

Menu.h

menu :

Fonction de menu.

Paramètre :

Nom	Type	Détail
library	library_t **	Pointeur sur la bibliothèque (par adresse)
borrowings	borrowings_t	Pointeur sur la liste des emprunts (par adresse)

Valeur de retour : void

menuChoice3 :

Action à exécuter lors du choix 3 dans le menu : lit le fichier des retours.

Paramètre :

Nom	Type	Détail
library	library_t *	Pointeur sur la bibliothèque
borrowings	borrowings_t **	Pointeur sur la liste des emprunts (par adresse)

Valeur de retour : void

menuChoice4 :

Action à exécuter lors du choix 4 dans le menu : affichage de la liste des emprunts avant le.... .

Paramètre :

Nom	Type	Détail
borrowings	borrowings_t *	Pointeur sur la liste des emprunts

Valeur de retour : void

remove_endstr_n :

Supprime le caractère \n de fin de chaîne de caractère s'il existe.

Paramètre :

Nom	Type	Détail
line	char *	Chaine de caractère

Valeur de retour : void

Traces

Nous détaillerons uniquement les traces des procédures qui modifient les structures.

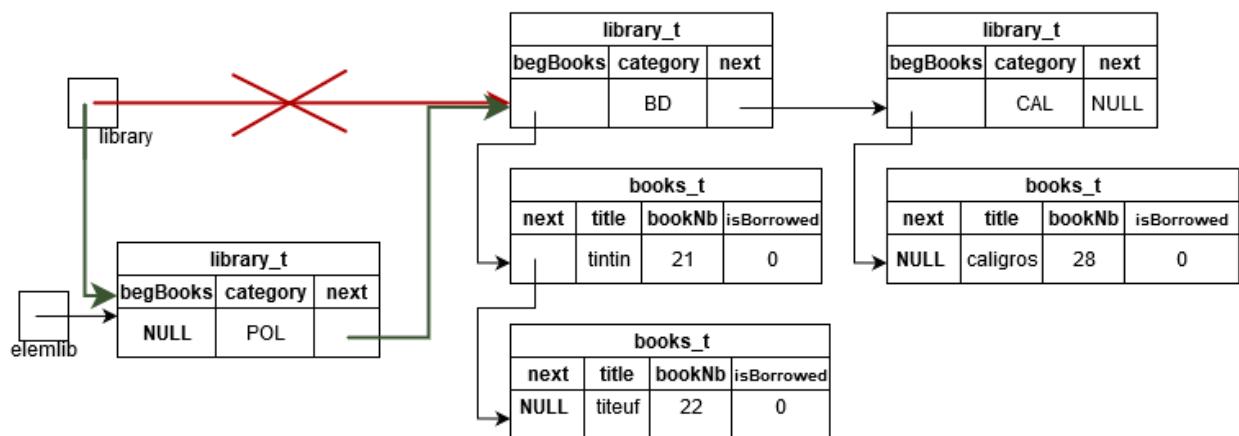
createCategory :

Créer une catégorie et l'ajoute dans la bibliothèque (en tête).

1-On crée la nouvelle categorie et on remplit les champs

library_t		
begBooks	category	next
NULL	POL	NULL

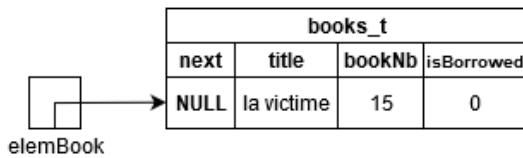
2-On insert en la nouvelle catégorie en tête de la bibliotheque



createBook :

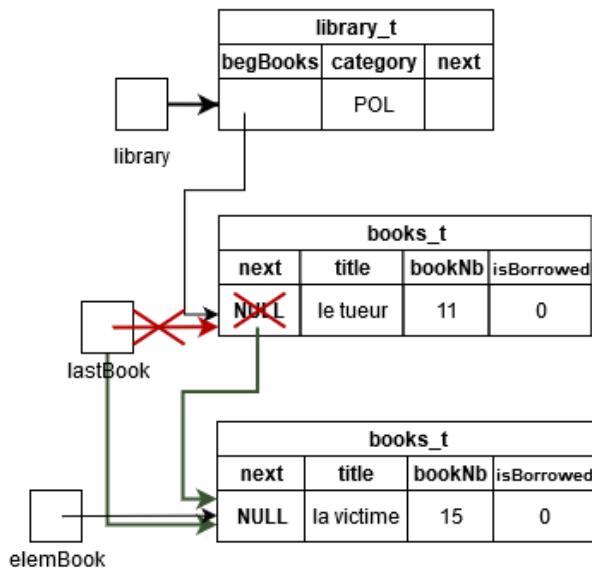
Créer un livre et l'ajoute à la fin de la liste des livres.

1-On crée le nouveau livre et on remplit les champs

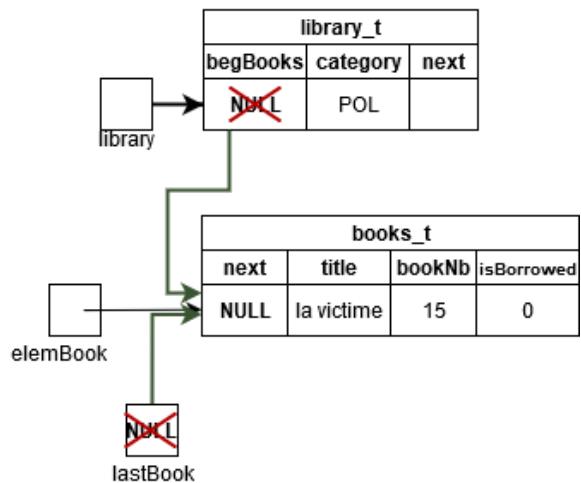


2-On ajoute le livre à la liste des livres de la catégorie (2 cas)

Si il y a déjà un livre dans la categorie



Si il n'y a pas de livre dans la catégorie



createLibrary :

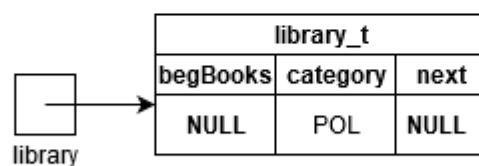
Créer la bibliothèque.

1-On part du pointeur sur la bibliothèque (passé en paramètre)



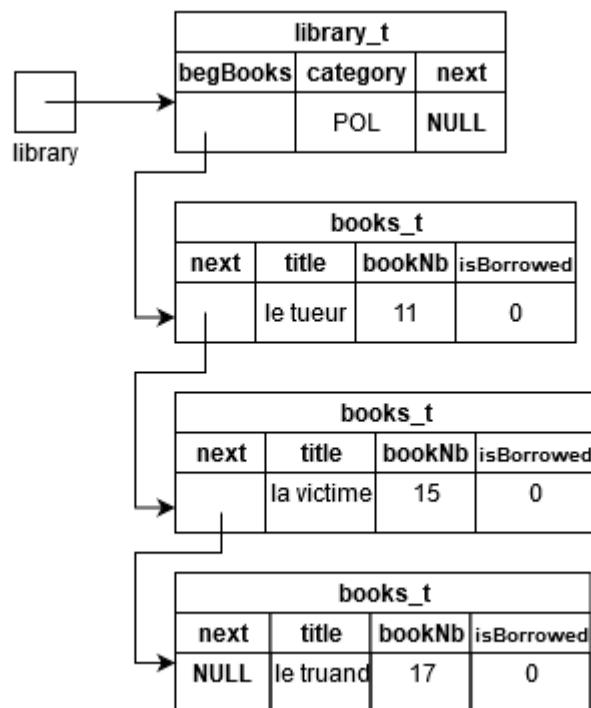
2-Tant que le fichier n'est pas lu complètement, on ajoute en tête la nouvelle catégorie

Appel procédure createCategory



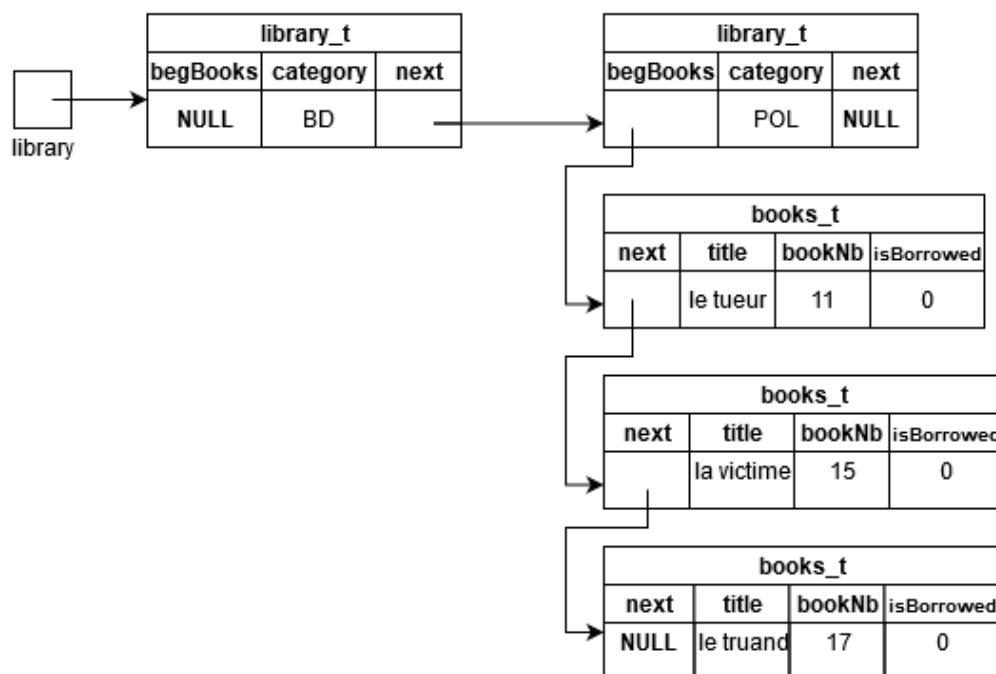
3-On ajoute les livres de la catégorie

Appel procédure createBook



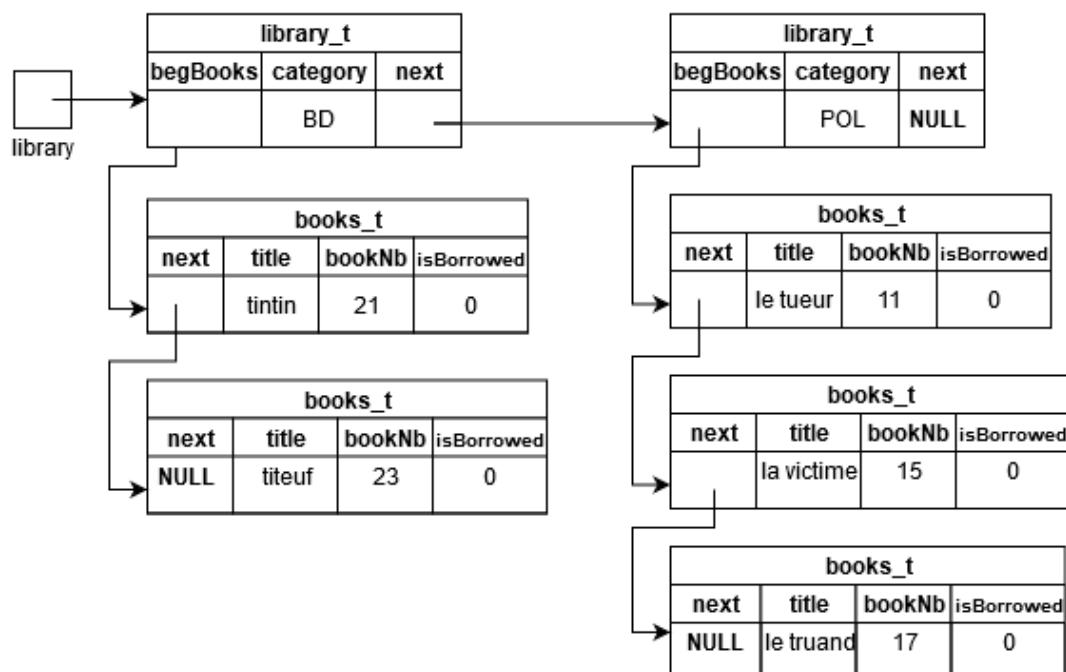
2-Tant que le fichier n'est pas vide, on ajoute en tête la nouvelle catégorie

Appel procédure createCategory



3-On ajoute les livres de la catégorie

Appel procédure createBook



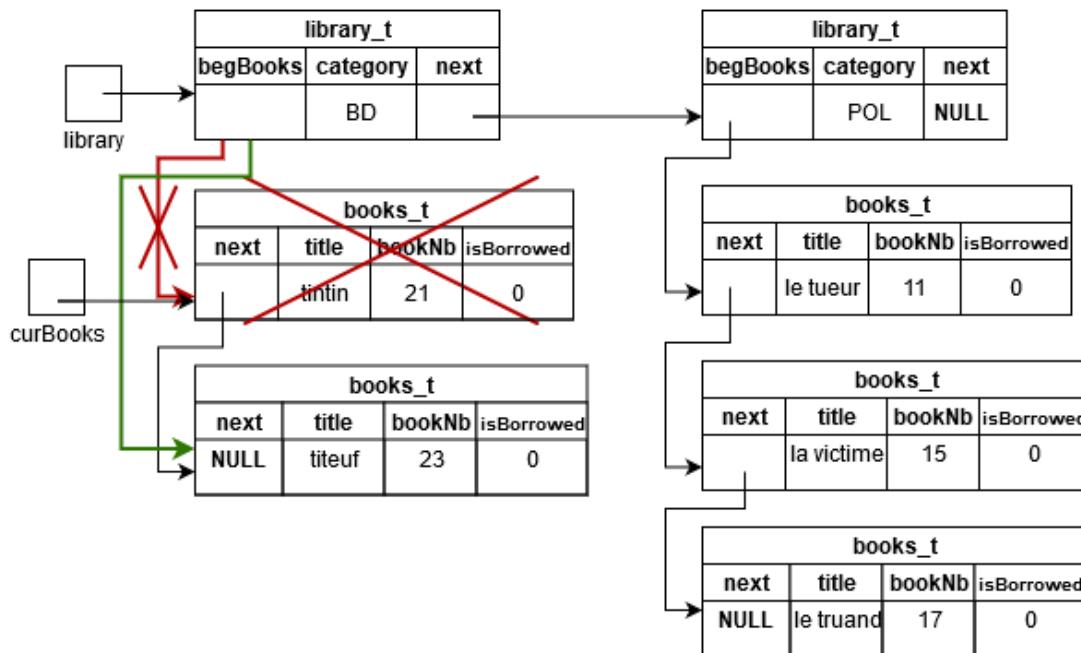
Et ainsi de suite jusqu'à avoir parcourue tout le fichier

freeLibrary :

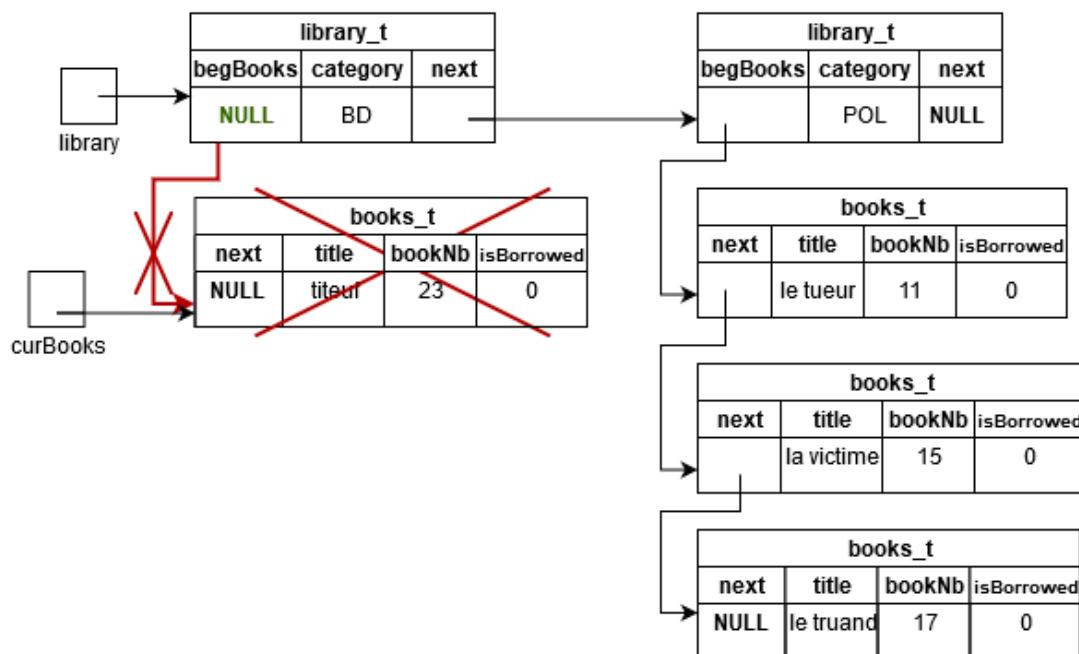
Libère la mémoire de la bibliothèque.

On supprime la première catégorie

On supprime le premier livre de cette catégorie

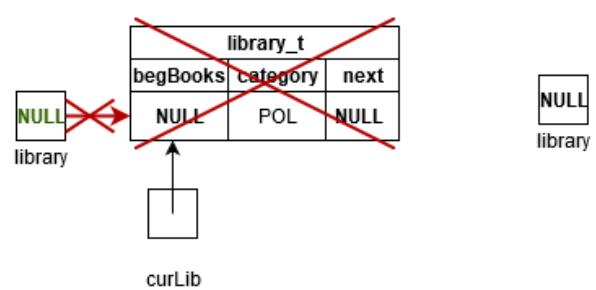
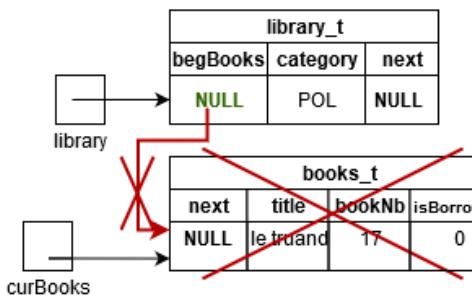
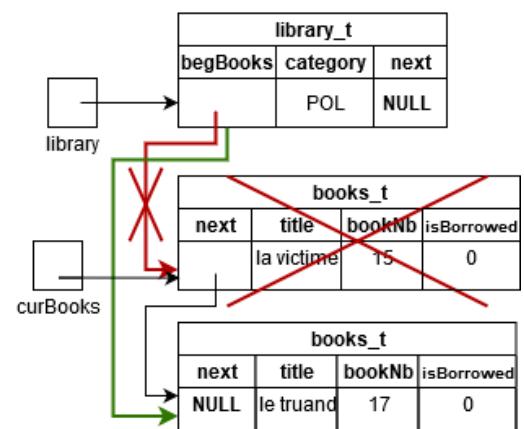
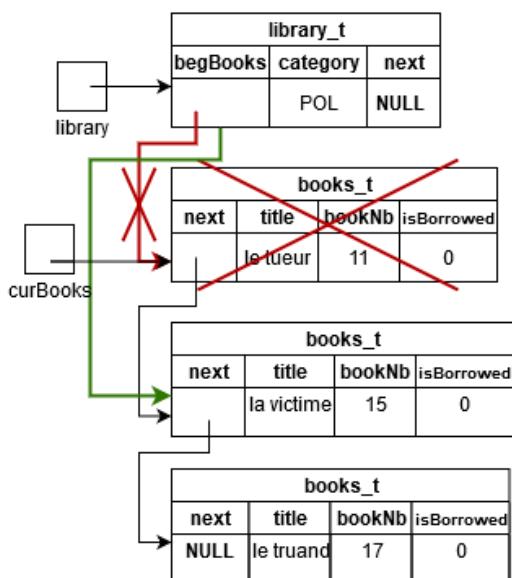
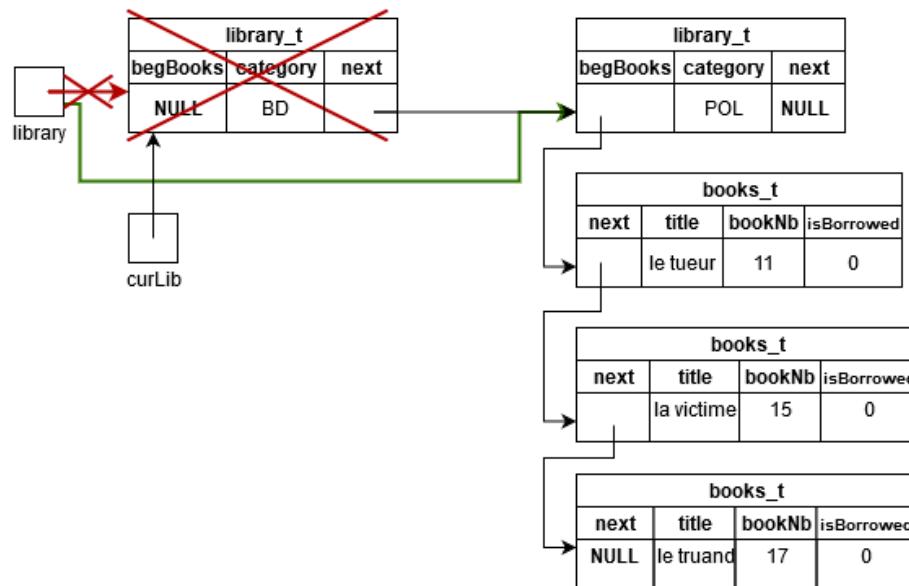


Tant qu'il y a des livres dans la catégories, on supprime celui qui est en tête



Tant qu'il y a des catégories dans la bibliothèque, on continue

Tant qu'il y a des catégories dans la bibliothèque, on continue



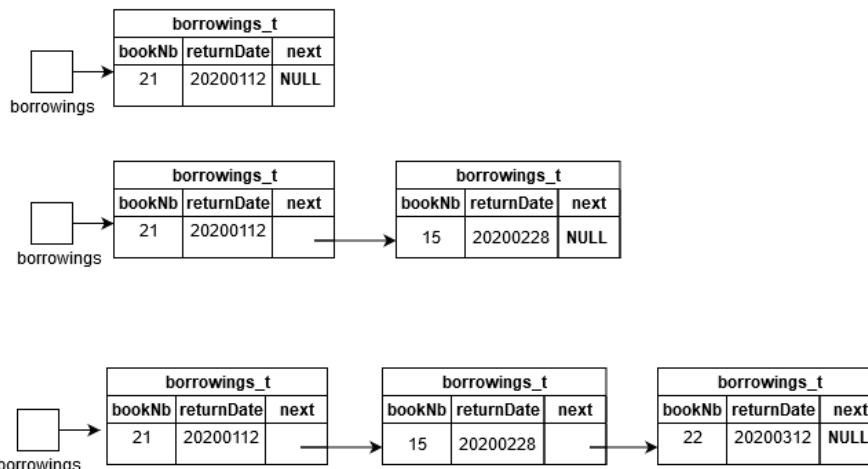
borrowBook :

Lit le fichier des emprunts et rempli la liste chaînée.

1-On part du pointeur sur la liste des emprunts

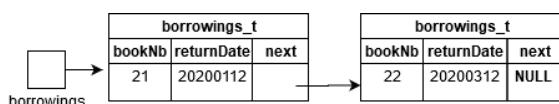


2-On insert les emprunts jusqu'a ce que le fichier soit vide
appel de la procédure insertBorrowing

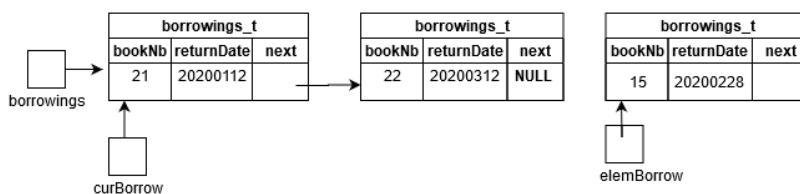


insertBorrowing :

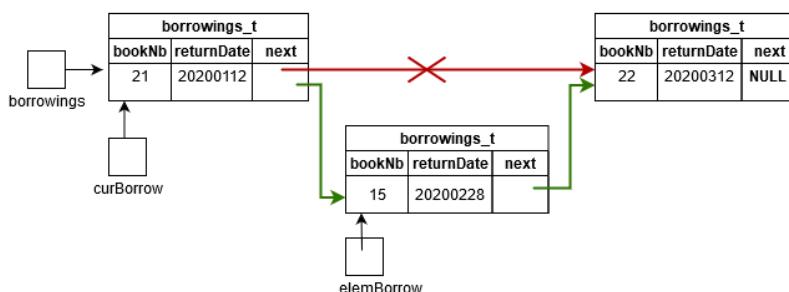
Insère le livre emprunté dans la liste des emprunts et modifie la valeur de isBorrowed du livre dans la bibliothèque.



On recherche l'élément précédent



On insère le nouvel élément



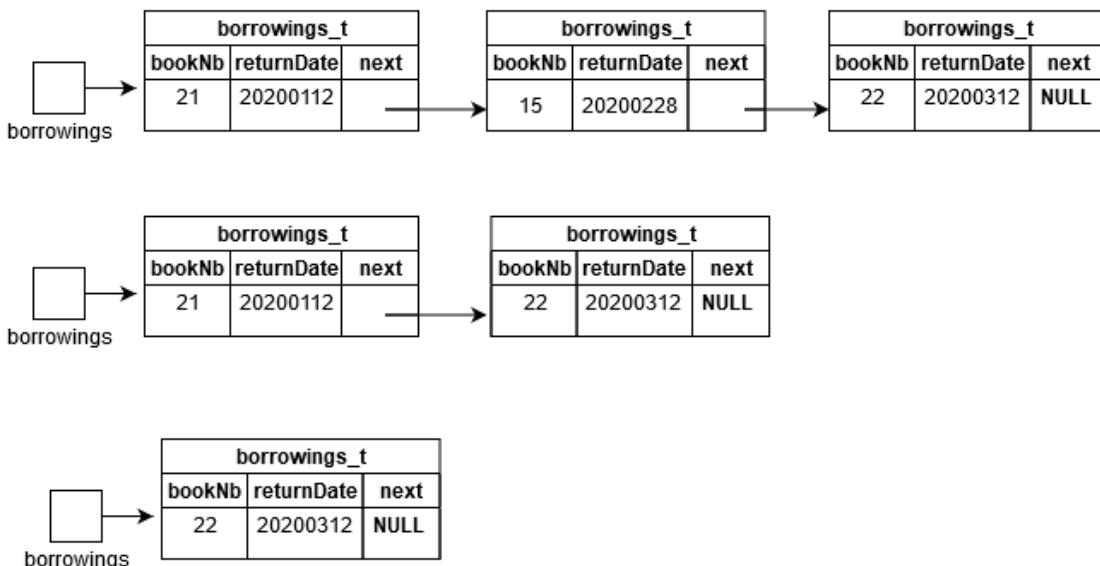
broughtBackBook :

Lit le fichier des retours, supprime les livres rendus de la liste des emprunts.

1-Tant que le fichier des retours n'est pas lu complètement,

on supprime les emprunts écrits dans le fichier

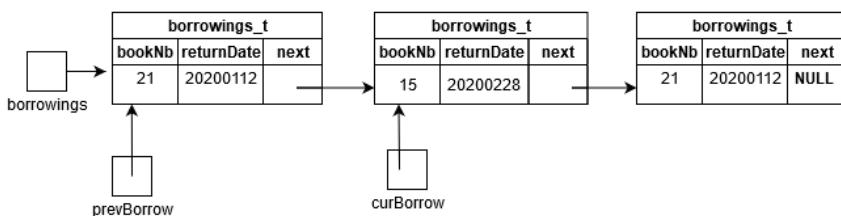
Appel de la procédure deleteBorrowing



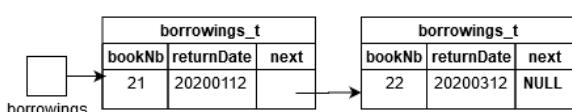
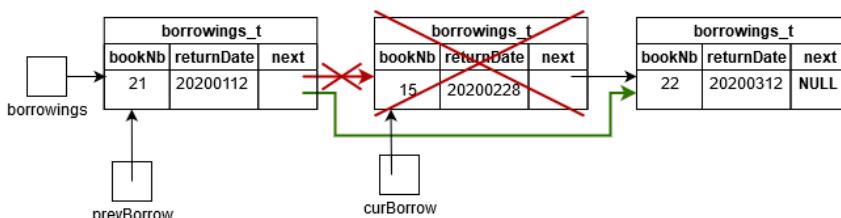
deleteBorrowing :

Supprime de la liste des emprunts le livre dont le numéro est passé en paramètre.

1-On cherche l'emprunt précédent celui à supprimer



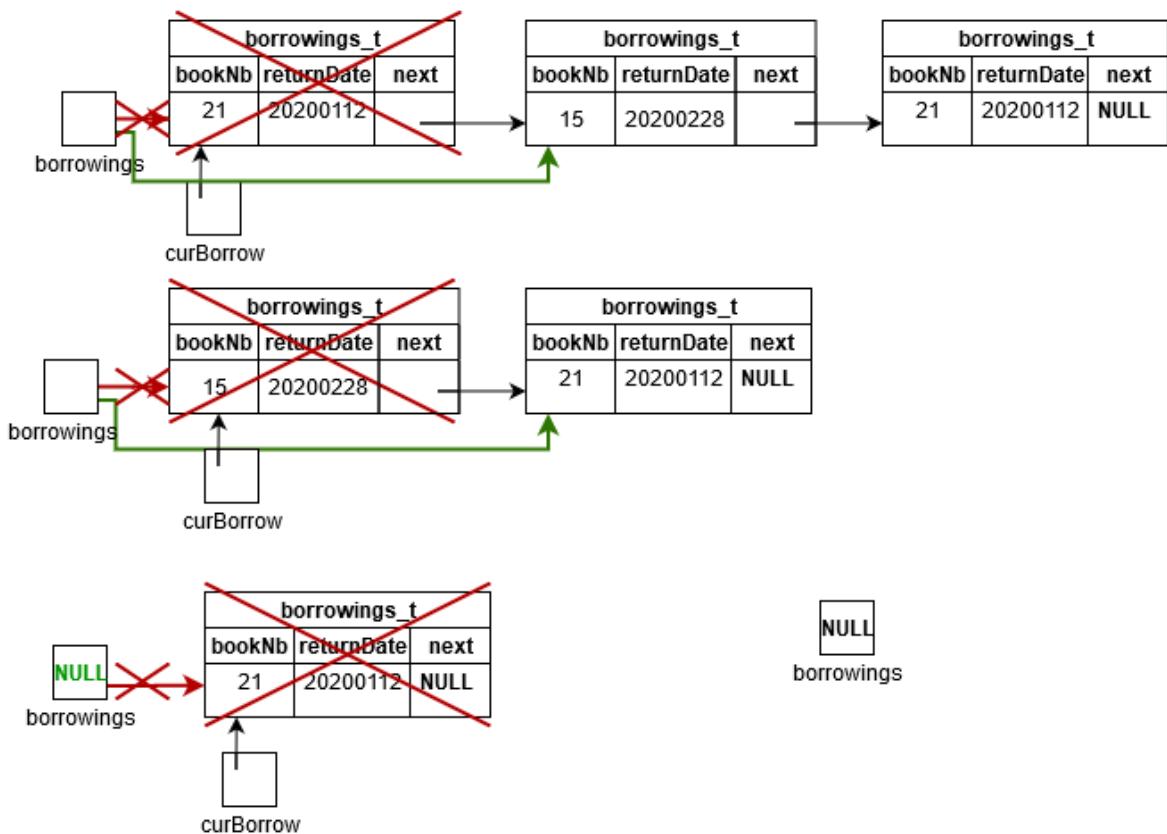
2-On supprime l'emprunt



freeBorrowings :

Libère la mémoire de la liste des emprunts.

Tant que la liste des emprunts n'est pas vide, on supprime celui en tête



Jeux d'essais, liste des cas

Cas General

library.txt	borrowings.txt	brought_back.txt
POL 3 11 le tueur 15 la victime 17 le truand BD 2 21 tintin 23 titeuf JAP 0 CAL 1 28 caligros	POL 15 20191112 BD 21 20200112 GHE 64 45213698 CAL 28 20180123	BD 21 CAL 28 JAP 12

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
  Oui  :1
  Non  :0
  -: 1
Création de la bibliothèque depuis : text_files/library.txt
Actualisation des livres empruntés depuis : text_files/borrowings.txt
Actualisation des livres rendus depuis : text_files/brought_back.txt

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter                      : 0
----- 1

On affiche la bibliothèque :
  CAL
    28 caligros 0
  JAP
  BD
    21 tintin 0
    23 titeuf 0
  POL
    11 le tueur 0
    15 la victime 1
    17 le truand 0

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter                      : 0
----- 2

On affiche la liste des emprunts :
  bookNb: 15  returnDate: 20191112
```

Cas Erreur

library.txt	borrowings.txt	brought_back.txt
POL 3 11 le tueur 15 la victime 17 le truand BD 2 21 tintin 23 titeuf JAP 0 CAL 1 28 caligros	GHE 64 45213698 BLO 00 00000000	GOU 93 JAP 12 JAP 12

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
  Oui :1
  Non :0
  -: 1
Création de la bibliothèque depuis : text_files/library.txt
Actualisation des livres empruntés depuis : text_files/borrowings.txt
Actualisation des livres rendus depuis : text_files/brought_back.txt

-----[ MENU ]-----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter                      : 0
-----[ -: 1 ]-----[ MENU ]-----
On affiche la bibliothèque :
  CAL
    28 caligros 0
  JAP
  BD
    21 tintin 0
    23 titeuf 0
  POL
    11 le tueur 0
    15 la victime 0
    17 le truand 0

-----[ MENU ]-----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter                      : 0
-----[ -: 2 ]-----[ MENU ]-----
On affiche la liste des emprunts :
Liste emprunts vide
```

Tout Empruntés et Tout Rendus

library.txt	borrowings.txt	brought_back.txt
POL 3	POL 15 20191112	BD 21
11 le tueur	BD 21 20200112	CAL 28
15 la victime	GHE 64 45213698	BD 23
17 le truand	CAL 28 20180123	JAP 12
BD 2	POL 11 33333333	POL 11
21 tintin	POL 17 22222222	POL 15
23 titeuf		POL 17
JAP 0		
CAL 1		
28 caligros		

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
 Oui :1
 Non :0
 -: 1
Création de la bibliothèque depuis : text_files/library.txt
Actualisation des livres empruntés depuis : text_files/borrowings.txt
Actualisation des livres rendus depuis : text_files/brought_back.txt

----- MENU -----
Afficher la bibliothèque : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter : 0
----- 1

On affiche la bibliothèque :
CAL
 28 caligros 0
JAP
BD
 21 tintin 0
 23 titeuf 0
POL
 11 le tueur 0
 15 la victime 0
 17 le truand 0

----- MENU -----
Afficher la bibliothèque : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter : 0
----- 2

On affiche la liste des emprunts :
Liste emprunts vide
```

Tout Empruntés et Rien Rendus

library.txt	borrowings.txt	brought_back.txt
POL 3	POL 15 20191112	
11 le tueur	BD 21 20200112	
15 la victime	GHE 64 45213698	
17 le truand	CAL 28 20180123	
BD 2	POL 11 33333333	
21 tintin	POL 17 22222222	
23 titeuf	BD 23 11111111	
JAP 0		
CAL 1		
28 caligros		

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
  Oui  :1
  Non  :0
    -: 1
Création de la bibliothèque depuis : text_files/library.txt
Actualisation des livres empruntés depuis : text_files/borrowings.txt
Actualisation des livres rendus depuis : text_files/brought_back.txt

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts a rendre avant le... : 4
Sauvegarder les emprunts dans un fichier   : 5
Quitter                         : 0
----: 1

On affiche la bibliothèque :
CAL
  28 caligros 1
JAP
BD
  21 tintin 1
  23 titeuf 1
POL
  11 le tueur 1
  15 la victime 1
  17 le truand 1

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts a rendre avant le... : 4
Sauvegarder les emprunts dans un fichier   : 5
Quitter                         : 0
----: 2

On affiche la liste des emprunts :
bookNb: 23  returnDate: 11111111
bookNb: 28  returnDate: 20180123
bookNb: 15  returnDate: 20191112
bookNb: 21  returnDate: 20200112
bookNb: 17  returnDate: 22222222
bookNb: 11  returnDate: 33333333
```

library vide

library.txt	borrowings.txt	brought_back.txt
	POL 15 20191112 BD 21 20200112 GHE 64 45213698 CAL 28 20180123	BD 21 CAL 28 JAP 12

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
Oui :1
Non :0
      -: 1
Création de la bibliothèque depuis : text_files/library.txt
Bibliothèque vide
```

borrowings vide

library.txt	borrowings.txt	brought_back.txt
POL 3 11 le tueur 15 la victime 17 le truand BD 2 21 tintin 23 titeuf JAP 0 CAL 1 28 caligros		BD 21 CAL 28 JAP 12

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
  Oui  :1
  Non  :0
      :- 1
Création de la bibliothèque depuis : text_files/library.txt
Actualisation des livres empruntés depuis : text_files/borrowings.txt
Actualisation des livres rendus depuis : text_files/brought_back.txt

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier   : 5
Quitter                         : 0
----- :- 1

On affiche la bibliothèque :
  CAL
    28 caligros 0
  JAP
  BD
    21 tintin 0
    23 titeuf 0
  POL
    11 le tueur 0
    15 la victime 0
    17 le truand 0

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier   : 5
Quitter                         : 0
----- :- 2

On affiche la liste des emprunts :
Liste emprunts vide
```

brought_back vide

library.txt	borrowings.txt	brought_back.txt
POL 3 11 le tueur 15 la victime 17 le truand BD 2 21 tintin 23 titeuf JAP 0 CAL 1 28 caligros	POL 15 20191112 BD 21 20200112 GHE 64 45213698 CAL 28 20180123	

```
| Créer la liste Bibliothèque et actualiser les livres empruntés ? |
  Oui :1
  Non :0
  -: 1
Création de la bibliothèque depuis : text_files/library.txt
Actualisation des livres empruntés depuis : text_files/borrowings.txt
Actualisation des livres rendus depuis : text_files/brought_back.txt

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter                      : 0
----- 1

On affiche la bibliothèque :
  CAL
    28 caligros 1
  JAP
  BD
    21 tintin 1
    23 titeuf 0
  POL
    11 le tueur 0
    15 la victime 1
    17 le truand 0

----- MENU -----
Afficher la bibliothèque      : 1
Afficher la liste des emprunts : 2
Lire le fichier des retours   : 3
Afficher les emprunts à rendre avant le... : 4
Sauvegarder les emprunts dans un fichier : 5
Quitter                      : 0
----- 2

On affiche la liste des emprunts :
  bookNb: 28  returnDate: 20180123
  bookNb: 15  returnDate: 20191112
  bookNb: 21  returnDate: 20200112
```

Annexes

makefile

```
◀ ▶ makefile x
1 #options
2 CFLAGS = -Wall -Wextra -g -MMD
3 LIB = -lm
4
5 SRC = $(wildcard src/*.c)
6 OBJ = $(patsubst src/%.c,build/%.o,$(SRC))
7 DEP = $(patsubst %.o,%.d,$(OBJ))
8
9 .PHONY: all
10 all: bin/executable
11     @echo "Exécuter avec ./bin/executable text_files/library.txt"
12     @mkdir -p "borrowings"
13
14 #édition des liens
15 bin/executable: $(OBJ)
16     @mkdir -p $(@D) #créer le dossier bin, s'il n'existe pas
17     gcc -o $@ $^ $(LIB)
18
19 #génère les fichiers .o (dans build) à partir des fichiers .c (dans src) du même nom
20 #comppilation
21 build/%.o: src/%.c
22     @mkdir -p $(@D) #créer le dossier build, s'il n'existe pas
23     gcc -c $< $(CFLAGS) -o $@
24
25 .PHONY: clean
26 clean:
27     rm -rf build
28
29 -include $(DEP)
30
```

structures

```

22  /* -----
23  *          Structure pour la liste des livres d'une catégorie
24  *          Liste non triée
25  * -----
26  typedef struct library {
27      char category[4];
28      struct books * begBooks;
29      struct library * next;
30 }library_t;
31

```

```

10  /* -----
11  *          Structure pour la liste des catégories (bibliothèque)
12  *          Triée selon l'entier bookNb croissant
13  * -----
14  typedef struct books {
15      int bookNb;
16      char title[11];
17      char isBorrowed;
18      struct books * next;
19 }books_t;
20

```

```

33  /* -----
34  *          Structure pour la liste des emprunts
35  *          Triée sur la date retour en croissant
36  * -----
37  typedef struct borrowings {
38      int bookNb;
39      char returnDate[9]; // Forme AAAAMMJJ
40      struct borrowings * next;
41 }borrowings_t;
42

```

main

```

main.c

1  /*
2   *          main.c
3   *          Contient le programme principal
4   */
5
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <string.h>
10 #include "common.h"
11 #include "borrow.h"
12 #include "lists.h"
13 #include "menu.h"
14
15
16 int main(int argc, char ** argv) {
17     if (argc<2) {
18         printf("Entrez un nom de fichier à lire\n");
19
20     } else {
21         library_t * library    = NULL; /*Bibliothèque*/
22         borrowings_t * borrowings = NULL; /*Liste emprunts*/
23
24         int           isStarted = 0;    /*On lance le programme ou non*/
25
26         char          filename[25];   /*Nom du fichier à appeler*/
27         char          filePath[40];  /*Chemin vers le fichier à appeler*/
28
29         printf("\n\033[33m | \033[36mCréer la liste Bibliothèque et actualiser les livres empruntés ?\033[33m |\n" \
30             "\t\033[32mOui :1\n\t\033[31mNon :0\033[00m\n\t\t- ");
31         scanf("%d",&isStarted);
32
33         /*Le programme se lance*/
34         if (isStarted == 1) {
35             /*Création de la bibliothèque depuis le fichier passé en paramètre lors de l'execution*/
36             createLibrary(argv[1], &library);
37
38             printf("  Crédation de la bibliothèque depuis : \033[32m%s\033[00m\n", argv[1]);
39
40             if (library != NULL) {
41                 /*Cherche s'il existe un fichier des emprunts dans le dossier borrowings*/
42                 /*et sélectionne le plus récent*/
43                 findFilenameMax(filename);
44
45                 /*Créer le bon filePath à partir de filename*/
46                 if (strcmp(filename, "0000-00-00_00h00'00'")) {
47                     snprintf(filePath, 40, "./borrowings/%s", filename);
48                 } else {
49                     strcpy(filePath, "text_files/borrowings.txt");
50                 }
51
52                 /*Création de la liste des emprunts à partir de filePath*/
53                 borrowBook(filePath, library, &borrowings);
54                 printf("  Actualisation des livres empruntés depuis : \033[32m%s\033[00m\n", filePath);
55
56                 strcpy(filePath, "text_files/brought_back.txt");
57                 /*Actualise les emprunts en fonction des retours à partir de filePath*/
58                 broughtBackBook(filePath, library, &borrowings);
59                 printf("  Actualisation des livres rendus depuis : \033[32m%s\033[00m\n", filePath);
60
61                 /*Lancement du menu*/
62                 menu(&library, &borrowings);
63
64             } else {
65                 printf("\033[31m  Bibliothèque vide\033[00m\n\n");
66             }
67
68             /*Libération de la mémoire*/
69             freeLibrary(&library);
70             freeBorrowings(&borrowings);
71
72         } else {
73             printf("\033[31m  Quitter\n\n\033[00m");
74         }
75     }
76
77     return 0;
}

```

```

11  /* -----
12  /* menu      Fonction de menu (graphique)
13  /*          On reste dans cette fonction tant qu'on ne choisi pas 0 */
14  /*
15  /* En entrée: library    : pointeur sur la bibliothèque (par adresse) */
16  /*           borrowings : pointeur sur la liste emprunts (par adresse) */
17  /*
18  /* En sortie: void
19  /* -----
20  void menu(library_t **, borrowings_t **);
21
22 void menu(library_t ** library, borrowings_t ** borrowings) {
23     int choice = -1; /*Choix dans le menu*/
24     int argNb = 0; /*Evite les bugs liés à une saisie incorrecte pour le choix du menu*/
25     char filename[255]; /*Pour stocker le nom du fichier à ouvrir*/
26
27     while (choice != 0) {
28
29         printf("\033[33m\n\n" \
30             " ----- \n" \
31             " |\033[35m           MENU           \033[33m|\n" \
32             " |\033[36m           Afficher la bibliothèque       : 1 \033[33m|\n" \
33             " |\033[36m           Afficher la liste des emprunts   : 2 \033[33m|\n" \
34             " |\033[36m           Lire le fichier des retours    : 3 \033[33m|\n" \
35             " |\033[36m           Afficher les emprunts à rendre avant le... : 4 \033[33m|\n" \
36             " |\033[36m           Sauvegarder les emprunts dans un fichier : 5 \033[33m|\n" \
37             " |\033[36m           Quitter                  : 0 \033[33m|\n" \
38             " ----- \n" \
39             "-: ");
40
41         // Dans un scanf : %n permet de récupérer le nombre de caractères lu par le scanf
42         // scanf return le nombre d'arguments qu'elle a rempli (l'argument %n n'est pas pris en compte)
43         argNb = scanf("%n%d", &argNb, &choice);
44         // On vide le buffer dans le cas où autre chose qu'un int a été entré
45         while (getchar () != '\n');
46
47         if (argNb == 1) {
48
49             switch (choice){
50
51                 case 0:
52                     printf("\n\033[31m  On quitte\033[00m\n\n");
53                     break;
54
55                 case 1:
56                     displayLibrary(*library);
57                     break;
58
59                 case 2:
60                     displayBorrowings(*borrowings);
61                     break;
62
63                 case 3:
64                     menuChoice3(*library, borrowings);
65                     break;
66
67                 case 4:
68                     menuChoice4(*borrowings);
69                     break;
70
71                 case 5:
72                     saveBorrowingsInFile(filename, *library, *borrowings);
73                     break;
74
75             } else {
76                 printf("\n\033[31m  Choix Invalide\033[00m\n");
77             }
78         }
79     }
80 }
```

```

23  /* -----
24  /* menuChoice3      Action à executer lors du choix 3 dans le menu      */
25  /*
26  /* En entrée: library      : pointeur sur la bibliothèque (par valeur)    */
27  /*                      borrowings : pointeur sur la liste emprunts (par adresse) */
28  /*
29  /* En sortie: void          :                                     */
30  /* ----- */
31 void menuChoice3(library_t *, borrowings_t **);
32

```

```

82 void menuChoice3(library_t * library, borrowings_t ** borrowings) {
83     char filename[255];           /*Pour stocker le nom du fichier à ouvrir*/
84     int typeFilename = 1;        /*Booléen -saisie du nom du fichier ou -nom automatique*/
85     char lineTmp[255];          /*Chaine de caractères pour récupérer la saisie avec fgets()*/
86
87     printf("\n" \
88         "\033[32m  |\033[36m Nom du fichier des rendus :      \033[32m|\n" \
89         "  |\033[36m      text_files/brought_back.txt : 1 \033[32m|\n" \
90         "  |\033[36m      Autre nom à entrer       : 0 \033[32m|\033[00m\n" \
91         "  -: ");
92     fgets(lineTmp, 255, stdin);
93     typeFilename = lineTmp[0] - 48;
94
95     if (typeFilename == 0) {
96         printf("\n\033[32m  |\033[36m Entrer le nom du fichier des retours : \033[32m|\033[00m\n" \
97             "  -: ");
98         fgets(filename, 255, stdin);
99         remove_endstr_n(filename);
100
101         printf("  Actualisation des livres empruntés depuis le fichier des retours :\n" \
102             "      \033[32m%s\033[00m\n", filename);
103         broughtBackBook(filename, library, borrowings);
104
105     } else if (typeFilename == 1) {
106         strcpy(filename, "text_files/brought_back.txt");
107         printf("  Actualisation des livres empruntés depuis le fichier des retours :\n" \
108             "      \033[32m%s\033[00m\n", filename);
109         broughtBackBook(filename, library, borrowings);
110
111     } else {
112         printf("\n\033[31m  Choix Invalide\033[00m\n");
113     }
114 }
115

```

```
34  /* ----- */  
35  /* menuChoice4      Action à executer lors du choix 4 dans le menu */  
36  /* */  
37  /* En entrée: borrowings : pointeur sur la liste emprunts (par valeur) */  
38  /* */  
39  /* En sortie: void */  
40  /* ----- */  
41 void menuChoice4(borrowings_t *);  
42
```

```
117 void menuChoice4(borrowings_t * borrowings) {  
118     char date[255]; /*date saisie*/  
119  
120     printf("\n" \  
121         "\033[32m | \033[36m Selectionner une date AAAAMMJJ : \033[32m|\n" \  
122         "\033[00m      -: ");  
123  
124     fgets(date, 255, stdin);  
125     remove_endstr_n(date);  
126  
127     if (isDateInputCorrect(date)) {  
128         displayBorrowingsBeforeDate(borrowings, date);  
129     } else {  
130         printf("\n\033[31m Date Invalide\033[00m\n");  
131     }  
132 }  
133 }
```

```
44  /* ----- */  
45  /* remove_endstr_n      Supprime le caractère \n de fin de chaîne */  
46  /*                      si il existe */  
47  /* */  
48  /* En entrée:  line : chaîne de caractères */  
49  /* */  
50  /* En sortie: void */  
51  /* ----- */  
52  void remove_endstr_n(char * line);
```

```
136 void remove_endstr_n(char * line){  
137     int i = 0; /*Compteur*/  
138  
139     while (line[i]!='\0') {  
140         i++;  
141     }  
142     i--;  
143  
144     if (line[i] == '\n') {  
145         line[i] = '\0';  
146     }  
147 }
```

```
65  /* ----- */  
66  /* isDateInputCorrect    Verifie si la chaine passée en paramètre est */  
67  /*                      une date de la forme AAAAMMJJ */  
68  /* */  
69  /* En entrée:  date : chaîne de caractères */  
70  /* */  
71  /* En sortie: int : 1 si réussi (format de la date conforme), 0 sinon */  
72  /* ----- */  
73  int isDateInputCorrect(char[9]);
```

```
150 int isDateInputCorrect(char date[255]) {  
151     int error = 1; /*Retour d'erreur*/  
152     int i = 0; /*Compteur*/  
153  
154     while (date[i] != '\0' && error == 1) {  
155         if (date[i] < 48 || date[i] > 57 || i >= 8) {  
156             error = 0;  
157         }  
158         i++;  
159     }  
160  
161     if (i != 8) {  
162         date[8] = '\0';  
163         error = 0;  
164     }  
165     return error;  
166 }  
167
```

```

76  /* -----
77  /* findFileName  Modifie la chaîne de caractères passées en paramètre,      */
78  /*                      Celle ci prend le nom du dernier fichier emprunts      */
79  /*                      créé, à lire lors de la prochaine exécution du programme */
80  /*          */
81  /* En entrée: filenameMax : chaîne de caractères                         */
82  /*          */
83  /* En sortie: void                                                       */
84  /*          */
85  void findFilenameMax(char [22]);
86

```

```

169 void findFilenameMax(char filenameMax[22]) {
170     DIR * rep = opendir("./borrowings"); /*Pointeur répertoire*/
171     strcpy(filenameMax, "0000-00-00_00h00'00''");
172
173     if (rep != NULL) {
174         struct dirent * ent = NULL; /*Pointeur entité*/
175         char filename[22]; /*Nom du fichier*/
176         char filenameDate[15]; /*Date de la forme AAAAMMJJhhmmss contenue dans le nom du fichier*/
177         char filenameDateMax[15] = "00000000000000";
178         int i = 0;
179         int j = 0;
180
181         while ((ent = readdir(rep)) != NULL) {
182
183             if (strcmp(ent->d_name, ".") && strcmp(ent->d_name, "..")) {
184
185                 strcpy(filename, ent->d_name);
186
187                 /*Remplissage de filenameDate avec uniquement les chiffres contenus dans filename*/
188                 i = 0; j = 0;
189                 while (filename[i] != '\0') {
190                     if (filename[i] >= 48 && filename[i] <= 57) {
191                         filenameDate[j] = filename[i];
192                         j++;
193                     }
194                     i++;
195                 }
196                 filenameDate[14] = '\0';
197
198                 /*filenameMax contient le nom du fichier le plus récent s'il existe, "0000-00-00_00h00'00'' sinon*/
199                 if (atoi(filenameDate) > atoi(filenameDateMax)) {
200                     strcpy(filenameMax, filename);
201                 }
202             }
203         }
204         closedir(rep);
205     }
206 }
207
208 }
209

```

List

```

11  /* -----
12  /* createLibrary  Creer la bibliothèque (liste chainée library) */
13  /*
14  /* En entrée: filename : nom du fichier à lire pour remplir la liste */
15  /*           library : pointeur sur la bibliothèque (par adresse) */
16  /*
17  /* En sortie: int : Retour d'erreur, 1 si réussi, 0 sinon */
18  /*
19  int createLibrary(char *, library_t **);
20

```

```

14 int createLibrary(char * filename, library_t ** library) {
15     FILE * file = NULL;          /*Fichier*/
16     file = fopen(filename, "r");
17
18     int      error = 1;          /*Retour d'erreur*/
19
20     char    category[4] = "_";   /*Catégorie lue dans le fichier*/
21     int      categorySize = 0;   /*Nombre de livre dans une catégorie*/
22
23     int      i = 0;              /*Compteur*/
24     books_t * lastBook = NULL;  /*Dernier livre créé*/
25
26     int      bookNb = 0;         /*Numéro du livre lu dans le fichier*/
27     char    title[11];          /*Titre lu dans le fichier*/
28
29     if (file != NULL) {
30
31         while (!feof(file) && error == 1) {
32
33             fscanf(file, "%s %d", category, &categorySize);
34
35             // vérification importante dans le cas où le fichier ne contient qu'une ligne avec \n
36             if (category[0] != '_') {
37                 error = createCategory(library, category);
38
39                 if (error == 1){
40
41                     for (i = 0; i < categorySize; i++) {
42                         // %[^\n] pour récupérer les chaînes de caractères comportants des espaces jusqu'au premier \n à l'aide d'un scanf
43                         fscanf(file, "%d %[^\n]", &bookNb, title);
44                         remove_endstr_r_windows(title);
45                         error = createBook(library, &lastBook, bookNb, title);
46                     }
47                 }
48             }
49         }
50         fclose(file);
51
52     } else {
53         error = 0;
54     }
55     return error;
56 }

```

```
22  /* ----- */  
23  /* createCategory Creer une categorie et l'ajoute dans la bibliothèque (en tête) */  
24  /* */  
25  /* En entrée: library : pointeur sur la bibliothèque (par adresse) */  
26  /*           category : nom de la catégorie en 3 caractères max */  
27  /* */  
28  /* En sortie: int : Retour d'erreur, 1 si réussi, 0 sinon */  
29  /* ----- */  
30  int createCategory (library_t **, char[4]);  
31
```

```
59  int createCategory(library_t ** library, char category[4]) {  
60      int          error = 1;      /*Retour d'erreur*/  
61      library_t * elemLib = (library_t *)malloc(sizeof(library_t));  
62  
63      if (elemLib != NULL) {  
64          strcpy(elemLib->category, category);  
65          elemLib->begBooks = NULL;  
66          elemLib->next = *library;  
67          *library = elemLib;  
68  
69      } else {  
70          error = 0;  
71      }  
72  
73      return error;  
74  }  
75
```

```
33  /* ----- */  
34  /* createBook Creer un livre et l'ajoute à la fin de la liste des livres */  
35  /* de la même catégorie */  
36  /* */  
37  /* En entrée: library : pointeur sur la bibliothèque (par adresse) */  
38  /* lastBook : pointeur sur le dernier livre de la categorie (par adresse) */  
39  /* bookNb : entier, numero du livre */  
40  /* title : chaine de caractères, nom du livre */  
41  /* */  
42  /* En sortie: int : Retour d'erreur, 1 si reussi, 0 sinon */  
43  /* ----- */  
44  int createBook(library_t **, books_t **, int, char[11]);  
45
```

```
77 ▼ int createBook(library_t ** library, books_t ** lastBook, int bookNb, char title[11]) {  
78     int error = 1; /*Retour d'erreur*/  
79     books_t * elemBook = (books_t *)malloc(sizeof(books_t));  
80  
81 ▼     if (elemBook != NULL) {  
82         elemBook->bookNb = bookNb;  
83         strcpy(elemBook->title, title);  
84         elemBook->isBorrowed = 0;  
85         elemBook->next = NULL;  
86  
87         if ((*library)->begBooks == NULL) {  
88             (*library)->begBooks = elemBook;  
89         } else {  
90             (*lastBook)->next = elemBook;  
91         }  
92  
93         *lastBook = elemBook;  
94     } else {  
95         error = 0;  
96     }  
97     return error;  
98 }
```

```
47  /* ----- */  
48  /* displayLibrary      Affiche la bibliothèque */  
49  /* */  
50  /* En entrée:  curLib : pointeur sur la bibliothèque (par valeur) */  
51  /* */  
52  /* En sortie:  void */  
53  /* ----- */  
54  void displayLibrary(library_t const *);  
55
```

```
116 void displayLibrary(library_t const * curLib) {  
117     books_t * curBooks = NULL; /*Pointeur courant sur la liste des livres d'une catégorie*/  
118  
119     printf("\n\033[32m    On affiche la bibliothèque :\033[00m\n");  
120  
121     if (curLib != NULL) {  
122         while (curLib != NULL) {  
123             printf("\t%s\n", curLib->category);  
124             curBooks = curLib->begBooks;  
125  
126             while (curBooks != NULL) {  
127                 printf("\t  %d %s %d\n", curBooks->bookNb, curBooks->title, curBooks->isBorrowed);  
128                 curBooks = curBooks->next;  
129             }  
130             curLib = curLib->next;  
131         }  
132     } else {  
133         printf("\n\033[31m    Liste bibliothèque vide\033[00m\n");  
134     }  
135 }  
136 }  
137 }  
138 }
```

```

57  /* -----
58  /* remove_endstr_r_windows  Supprime le caractère \r de fin de chaîne */
59  /*                               si il existe
60  */
61  /* En entrée:  line : chaîne de caractères
62  */
63  /* En sortie:  void
64  */
65  void remove_endstr_r_windows(char *);
66

```

```

102 void remove_endstr_r_windows(char * line){
103     int i = 0; /*Compteur*/
104
105     while (line[i]!='\0') {
106         i++;
107     }
108     i--;
109
110     if (line[i] == '\r') {
111         line[i] = '\0';
112     }
113 }
114

```

```

68  /* -----
69  /* freeLibrary           Libère la mémoire
70  */
71  /* En entrée:  library : pointeur sur la bibliothèque (par adresse)
72  */
73  /* En sortie:  void
74  */
75  void freeLibrary(library_t **);
76

```

```

140 void freeLibrary(library_t ** library) {
141     library_t * curLib = *library; /*Pointeur courant sur la bibliothèque*/
142     books_t * curBooks = NULL;    /*Pointeur courant sur la liste des livres d'une catégorie*/
143
144     /*Libération de la bibliothèque*/
145     while (*library != NULL) {
146         curBooks = curLib->begBooks;
147
148         /*Libération des livres d'une catégorie*/
149         while (curLib->begBooks != NULL) {
150             curBooks = curBooks->next;
151             free(curLib->begBooks);
152             curLib->begBooks = curBooks;
153         }
154
155         /*Libération des catégories*/
156         curLib = curLib->next;
157         free(*library);
158         *library = curLib;
159     }
160 }
161

```

Borrow

```

21  /* -----
22  /* borrowBook Lit le fichier des emprunts et rempli la liste chainée */
23  /*
24  /* En entrée: filename : chaîne de caractères, nom du fichier */
25  /*           library : pointeur sur la bibliothèque (par valeur) */
26  /*           borrowings : pointeur sur la liste emprunts (par adresse)*/
27  /*
28  /* En sortie: void */
29  /*
30  void borrowBook(char *, library_t *, borrowings_t **);
31

```

```

33 void borrowBook(char * filename, library_t * library, borrowings_t ** borrowings) {
34     FILE * file = NULL; /*Fichier*/
35     file = fopen(filename, "r");
36
37     books_t * bookBorrowed = NULL;
38
39     if (library != NULL) {
40
41         if (file != NULL) {
42             char category[4] = " "; /*Categorie lue dans le fichier*/
43             int bookNb = 0; /*Numero du livre lu dans le fichier*/
44             char date[9]; /*Date de rendu lue dans le fichier*/
45
46             while (!feof(file)) {
47                 fscanf(file, "%s %d %s", category, &bookNb, date);
48
49                 // Vérification importante dans le cas où le fichier ne contient qu'une ligne avec \n
50                 if (category[0] != '.') {
51                     bookBorrowed = isBookInLibrary(library, category, bookNb); /*bookBorrowed == NULL si le livre n'a pas été trouvé dans la bibliothèque*/
52                     /*On insère le livre dans la liste des emprunts*/
53                     insertBorrowing(borrowings, bookBorrowed, date);
54                 }
55             }
56             fclose(file);
57
58         } else {
59             printf("\n\033[31m Nom de fichier pour les emprunts inexistant\033[00m\n");
60         }
61
62     } else {
63         printf("\n\033[31m Bibliothèque vide\033[00m\n");
64     }
65 }
66

```

```
33  /* ----- */  
34  /* isBookInLibrary Cherche dans la bibliothèque si le livre */  
35  /* que l'on veut emprunter existe */  
36  /* */  
37  /* En entrée: curlib : pointeur courant sur la bibliothèque (par valeur) */  
38  /* category : catégorie du livre recherché */  
39  /* bookNb : numéro du livre recherché */  
40  /* */  
41  /* En sortie: pointeur contenant l'adresse du livre de la bibliothèque */  
42  /* que l'on souhaite emprunter, NULL si il n'est pas trouvé */  
43  /* ----- */  
44 books_t * isBookInLibrary(library_t const *, char[4], int);  
  
68 books_t * isBookInLibrary(library_t const * curlib, char category[4], int bookNb) {  
69     books_t * curBooks = NULL; /*Pointeur courant sur la liste des livres d'une catégorie*/  
70  
71     while (curlib != NULL && strcmp(curlib->category, category)) {  
72         curlib = curlib->next;  
73     }  
74  
75     if (curlib != NULL) {  
76         curBooks = curlib->begBooks;  
77         while (curBooks != NULL && curBooks->bookNb != bookNb) {  
78             curBooks = curBooks->next;  
79         }  
80     }  
81     return curBooks;  
82 }  
83 }
```

```

47  /* -----
48  /* insertBorrowing    Insere le livre emprunté dans la liste emprunts et      */
49  /*                      modifie la valeur de isBorrowed dans la bibliothèque   */
50  /*                      */
51  /* En entrée: borrowings : pointeur sur la liste emprunts (par adresse)  */
52  /*             bookBorrowed : adresse du livre de la bibliothèque que l'on   */
53  /*                         souhaite emprunter                                */
54  /*             date          : chaîne de caractère, date retour du livre   */
55  /*             */
56  /* En sortie: void           */
57  /* ----- */
58 void insertBorrowing(borrowings_t **, books_t * curBooks, char[9]);
59

```

```

86 void insertBorrowing(borrowings_t ** borrowings, books_t * bookBorrowed, char date[9]) {
87     borrowings_t * curBorrow = *borrowings; /*Pointeur courant sur la liste des emprunts*/
88     borrowings_t * elemBorrow = NULL;           /*Element alloué*/
89
90     /*Modifie la valeur de isBorrowed dans la bibliothèque*/
91     if (bookBorrowed != NULL && bookBorrowed->isBorrowed != 1) {
92         bookBorrowed->isBorrowed = 1;
93
94         elemBorrow = (borrowings_t *)malloc(sizeof(borrowings_t));
95
96         if (elemBorrow != NULL) {
97             elemBorrow->bookNb = bookBorrowed->bookNb;
98             strcpy(elemBorrow->returnDate, date);
99
100            /*Place le livre au bon endroit dans la bibliothèque (trié par date croissante)*/
101            if (curBorrow != NULL && atoi(date) > atoi(curBorrow->returnDate)) {
102                while (curBorrow->next != NULL && atoi(date) > atoi(curBorrow->next->returnDate)) {
103                    curBorrow=curBorrow->next;
104                }
105                elemBorrow->next = curBorrow->next;
106                curBorrow->next = elemBorrow;
107            } else {
108                *borrowings = elemBorrow;
109                elemBorrow->next = curBorrow;
110            }
111        }
112
113        /*else : Livre de la liste à emprunter inexistant*/
114    }
115 }

```

```

62  /* ----- */
63  /* broughtBackBook Lit le fichier des retours, supprime les livres */
64  /* rendus de la liste des emprunts et modifie la */
65  /* valeur de isBorrowed dans la bibliothèque */
66  /*
67  /* En entrée: filename : chaîne de caractère, nom du fichier des retours*/
68  /* library : pointeur sur la bibliothèque (par valeur) */
69  /* borrowings : pointeur sur la liste emprunts (par adresse) */
70  /*
71  /* En sortie: void */
72  /*
73  void broughtBackBook(char *, library_t *, borrowings_t **);
74

```

```

118 void broughtBackBook(char * filename, library_t * library, borrowings_t ** borrowings) {
119     FILE * file = NULL; /*Fichier*/
120     file = fopen(filename, "r");
121
122     if (library != NULL) {
123
124         if (file != NULL) {
125             char category[4] = "_"; /*Categorie lue dans le fichier*/
126             int bookNb = 0; /*Numero du livre lu dans le fichier*/
127
128             while (!feof(file)) {
129                 fscanf(file, "%s %d", category, &bookNb);
130
131                 // Vérification importante dans le cas où le fichier ne contient qu'une ligne avec \n
132                 if (category[0] != '_') {
133                     /*Supprime le livre de la liste emprunts*/
134                     deleteBorrowing(borrowings, bookNb);
135                     /*Modifie la valeur de isBorrowed dans la bibliothèque*/
136                     isBorrowedToFalse(library, category, bookNb);
137                 }
138             }
139
140             fclose(file);
141         } else {
142             printf("\n\033[31m Nom de fichier pour les retours inexistant\033[00m\n");
143         }
144
145     } else {
146         printf("\n\033[31m Liste bibliothèque vide\033[00m\n");
147     }
148 }

```

```
76  /* ----- */  
77  /* deleteBorrowing Supprime de la liste des emprunts le livre dont */  
78  /*                      le numéro est passé en paramètre (libération mémoire) */  
79  /* */  
80  /* En entrée: borrowings : pointeur sur la liste emprunts (par adresse) */  
81  /*          bookNb      : entier, numéro du livre à supprimer */  
82  /* */  
83  /* En sortie: void */  
84  /* ----- */  
85  void deleteBorrowing(borrowings_t ** , int);  
86
```

```
151 void deleteBorrowing(borrowings_t ** borrowings, int bookNb) {  
152     borrowings_t * curBorrow = *borrowings; /*Pointeur courant sur la liste des emprunts*/  
153     borrowings_t * prevBorrow = *borrowings; /*Pointeur courant précédent sur la liste des emprunts*/  
154  
155     if (*borrowings != NULL) {  
156         while (curBorrow != NULL && curBorrow->bookNb != bookNb) {  
157             prevBorrow = curBorrow;  
158             curBorrow = curBorrow->next;  
159         }  
160     }  
161     if (curBorrow != NULL) {  
162         prevBorrow->next = curBorrow->next;  
163     } else {  
164         prevBorrow->next = NULL;  
165     }  
166     if (curBorrow == *borrowings) {  
167         *borrowings = (*borrowings)->next;  
168     }  
169     free(curBorrow);  
170 }  
171 }
```

```
88  /* ----- */
89  /* isBorrowedToFalse   Passe la valeur de isBorrowed de la bibliothèque */
90  /*                      à False (le livre a été rendu) */
91  /*
92  /* En entrée:  curLib    : pointeur courant sur la liste bibliothèque
93  /*                  category : chaîne de caractères, catégorie du livre rendu
94  /*                  bookNb   : entier, numéro du livre rendu
95  /*
96  /* En sortie: void
97  /* ----- */
98 void isBorrowedToFalse(library_t *, char[4], int);
99
```

```
177 void isBorrowedToFalse(library_t * curLib, char category[4], int bookNb) {
178     /*curBooks : Pointeur courant sur la liste des livres d'une catégorie*/
179
180     /*Cherche le livre rendu dans la bibliothèque*/
181     books_t * curBooks = isBookInLibrary(curLib, category, bookNb);
182
183     /*Modifie la valeur de isBorrowed dans la bibliothèque*/
184     if (curBooks != NULL && curBooks->bookNb == bookNb) {
185         curBooks->isBorrowed = 0;
186     }
187 }
```

```

102
103  /* -----
104  * displayBorrowingsBeforeDate  Affiche le numéro et dateRetour des
105  *                               livres à rendre avant date (en paramètre)
106  */
107  /* En entrée:  curBorrow : pointeur courant sur la liste emprunts (par valeur) */
108  /*           date      : chaîne de caractère
109  */
110  /* En sortie: void
111  */
112  void displayBorrowingsBeforeDate(borrowings_t const *, char[9]);
113

```

```

190 void displayBorrowingsBeforeDate(borrowings_t const * curBorrow, char date[9]) {
191     int i = 0; /*Compteur*/
192
193     printf("\n");
194     while (curBorrow != NULL && atoi(curBorrow->returnDate) <= atoi(date)) {
195         printf("    Livre numero : \033[35m%d\033[00m à rendre avant le \033[35m%s\033[00m\n", curBorrow->bookNb, curBorrow->returnDate);
196         curBorrow = curBorrow->next;
197         i++;
198     }
199
200     if (i == 0) {
201         printf("    Aucun livre à rendre avant cette date\n");
202     }
203 }

```

```

11  /* -----
12  * displayBorrowings      Affiche la liste des emprunts
13  */
14  /* En entrée:  curBorrow : pointeur courant sur la liste emprunts (par valeur) */
15  */
16  /* En sortie: void
17  */
18  void displayBorrowings(borrowings_t const *);
19

```

```

18 void displayBorrowings(borrowings_t const * curBorrow) {
19     printf("\n\033[32m  On affiche la liste des emprunts :\033[00m\n");
20
21     if (curBorrow != NULL) {
22         while (curBorrow != NULL) {
23             printf("\tbookNb: \033[35m%d\033[00m  returnDate: \033[35m%s\033[00m\n", curBorrow->bookNb, curBorrow->returnDate);
24             curBorrow = curBorrow->next;
25         }
26     } else {
27         printf("\n\033[31m  Liste emprunts vide\033[00m\n");
28     }
29
30 }
31

```

```
115  /* ----- */
116  /* createFilename Remplie la chaîne de caractères passée en paramètre */
117  /*           avec la date et l'heure à laquelle est créé le fichier */
118  /*
119  /* En entrée: filename : chaîne de caractères
120  /*
121  /* En sortie: void
122  /*
123  void createFilename(char *);
```

```
206 void createFilename(char * filename) {
207     int h, min, s, day, month, year;
208     time_t now;
209
210     time(&now);
211
212     mkdir("borrowings", 777);
213
214     struct tm *local = localtime(&now);
215     h      = local->tm_hour;
216     min    = local->tm_min;
217     s      = local->tm_sec;
218     day    = local->tm_mday;
219     month  = local->tm_mon + 1;
220     year   = local->tm_year + 1900;
221
222     sprintf(filename, 40, "borrowings/%d-%02d-%02d_%02dh%02d'%02d'", year, month, day, h, min, s);
223 }
```

```
126  /* ----- */  
127  /* saveBorrowingsInFile Sauvegarde les emprunts dans un fichier pouvant */  
128  /*                      être lu par borrowBook() */  
129  /* */  
130  /* En entrée: filename : chaîne de caractères, nom du fichier */  
131  /*          library : pointeur sur la bibliothèque (par valeur) */  
132  /*          curBorrow : pointeur courant sur la liste des emprunts */  
133  /* */  
134  /* En sortie: void */  
135  /* ----- */  
136  void saveBorrowingsInFile(char *, library_t const *, borrowings_t const * curBorrow);  
137
```

```
226 void saveBorrowingsInFile(char * filename, library_t const * library, borrowings_t const * curBorrow) {  
227     createFilename(filename);  
228  
229     FILE * file = NULL; /*Fichier*/  
230     file = fopen(filename, "w");  
231     char category[4]; /*Categorie du livre à sauvegarder*/  
232  
233     if (file != NULL) {  
234         printf("\n    fichier créé : \033[35m%s\033[00m\n", filename);  
235  
236         while (curBorrow != NULL) {  
237             findCategoryName(library, curBorrow->bookNb, category);  
238             fprintf(stdout, "%s %d %s\n", category, curBorrow->bookNb, curBorrow->returnDate);  
239             fprintf(file, "%s %d %s", category, curBorrow->bookNb, curBorrow->returnDate);  
240             curBorrow = curBorrow->next;  
241  
242             if (curBorrow != NULL) {  
243                 fprintf(file, "\n");  
244             }  
245         }  
246         fclose(file);  
247     }  
248 }
```

```
139  /* ----- */
140  /* findCategoryName Remplie la chaîne de caractères passée en paramètre */
141  /*           avec le nom de la catégorie correspondant au numéro */
142  /*           du livre */
143  /*
144  /* En entrée: curLib : pointeur courant sur la liste bibliothèque
145  /*           bookNb : entier, numéro du livre
146  /*           category : chaîne de caractères à remplir
147  /*
148  /* En sortie: void
149  /* ----- */
150 void findCategoryName(library_t const *, int, char[4]);
151
```

```
253 void findCategoryName(library_t const * curLib, int bookNb, char category[4]) {
254     int isfound = 0; /*Livre trouvé (1) ou non (0)*/
255     books_t * curBooks = NULL; /*Pointeur courant sur la liste des livres d'une catégorie*/
256
257     while (curLib != NULL && isfound == 0) {
258         strcpy(category, curLib->category);
259         curBooks = curLib->begBooks;
260
261         while (curBooks != NULL && curBooks->bookNb != bookNb) {
262             curBooks = curBooks->next;
263         }
264
265         if (curBooks != NULL) {
266             isfound = 1;
267         }
268
269     }
270 }
271 }
```

```
153  /* ----- */  
154  /* freeBorrowings      Libère la mémoire  
155  */  
156  /* En entrée: borrowings : pointeur sur la liste emprunts (par adresse) */  
157  /* */  
158  /* En sortie: void */  
159  /* ----- */  
160 void freeBorrowings(borrowings_t ** borrowings);  
161
```

```
274 void freeBorrowings(borrowings_t ** borrowings) {  
275     borrowings_t * curBorrow = *borrowings; /*Pointeur courant sur la liste des emprunts*/  
276  
277     /*Libération de la liste des emprunts*/  
278     while (*borrowings != NULL) {  
279         curBorrow = curBorrow->next;  
280         free(*borrowings);  
281         *borrowings = curBorrow;  
282     }  
283 }  
284
```