Rapport
Structure de donnée
TP 1

Encadrant : Daniel Yves Jean Mars 2020

Sommaire

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

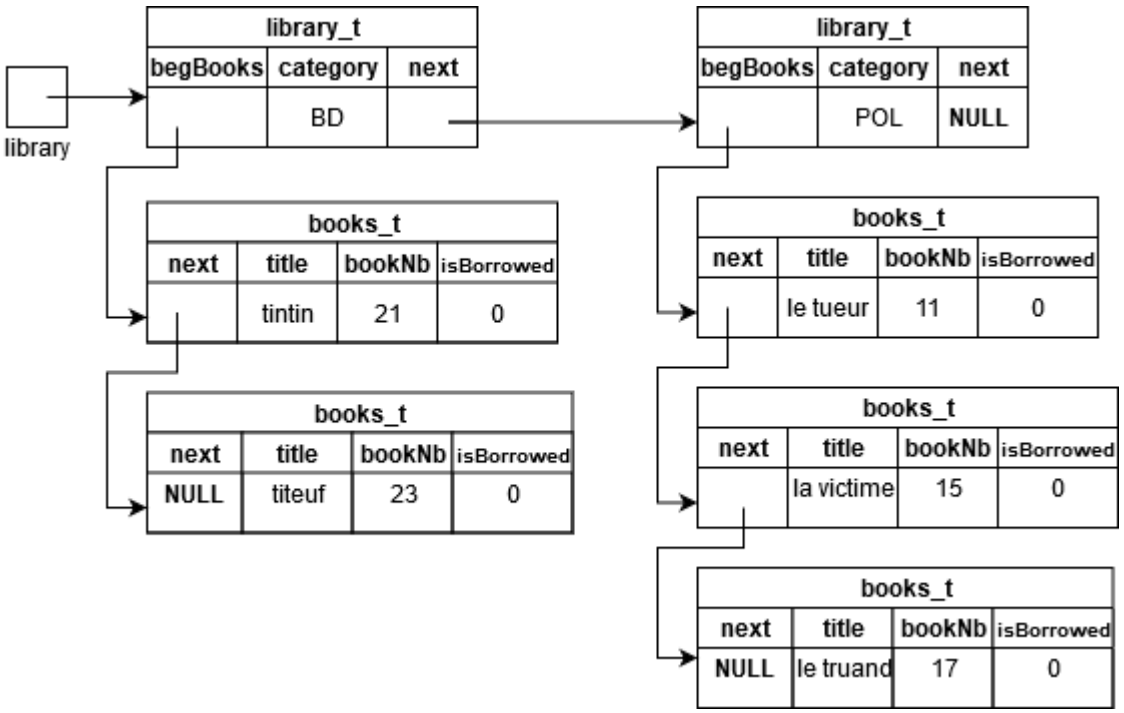


Schéma d'un exemple de structure pour library

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

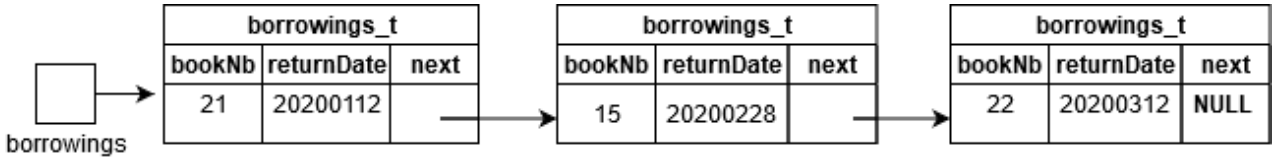


Schéma d'un exemple de structure pour borrowings

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

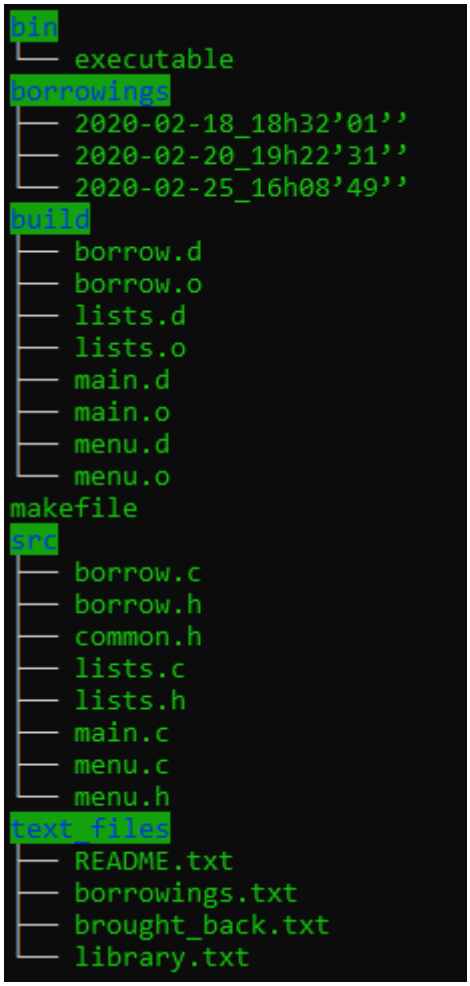
Tous les livres rendus sont enregistrés dans un fichier permettant d'actualiser l'état des livres rendus et modifier la liste des livres empruntés.

Structure :
<catégorie du livre> <numéro du livre>

Exemple :

```
BD 21
CAL 28
```

Arborescence du projet



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.
c**lean** : 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.

Variables :

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.

Variables internes :

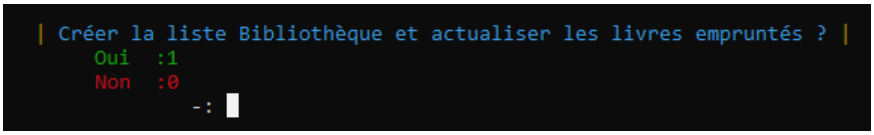
$@ : Le nom de la cible.
$< : Le nom de la première dépendance.
$^ : La liste des dépendances.
@ : Permet de ne pas afficher la commande dans la console.

Mode d'emploi

Le programme fonctionne de la manière suivante :

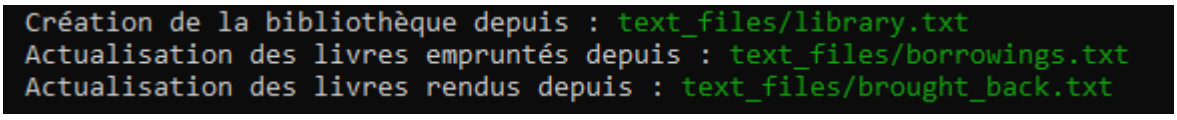
Exécuter le programme : bin/executable text_files/library.txt

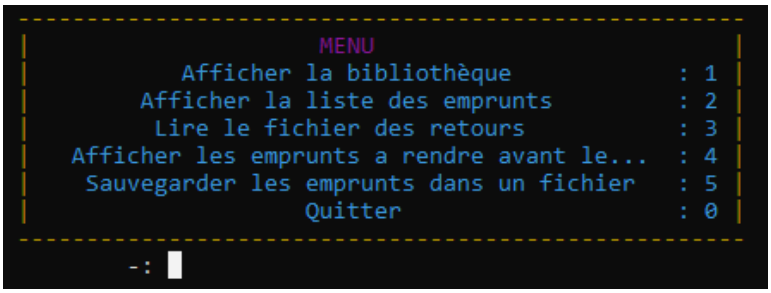
Une fois l'exécutable lancé, le programme effectue une vérification du fichier Livres qui va être lu.



Entrer 1 pour lire le fichier, 0 pour quitter.

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

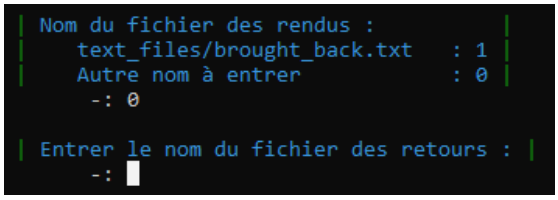




Vient ensuite le menu, où l'utilisateur peut effectuer des actions.
Voici les options que l'utilisateur peut effectuer :

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 :



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 AAAAMMJJ : |
      -:
```

L'utilisateur doit alors entrer une date de la forme AAAAMMJJ 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 chaine 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 le liste chainée.

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 AAAAMMJJ)

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_hhHmm'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 chaine de caractère s'il existe.

Paramètre :

Nom	Type	Détail
line	char *	Chaine de caractère

Valeur de retour : void

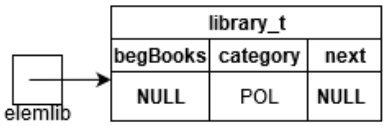
Traces

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

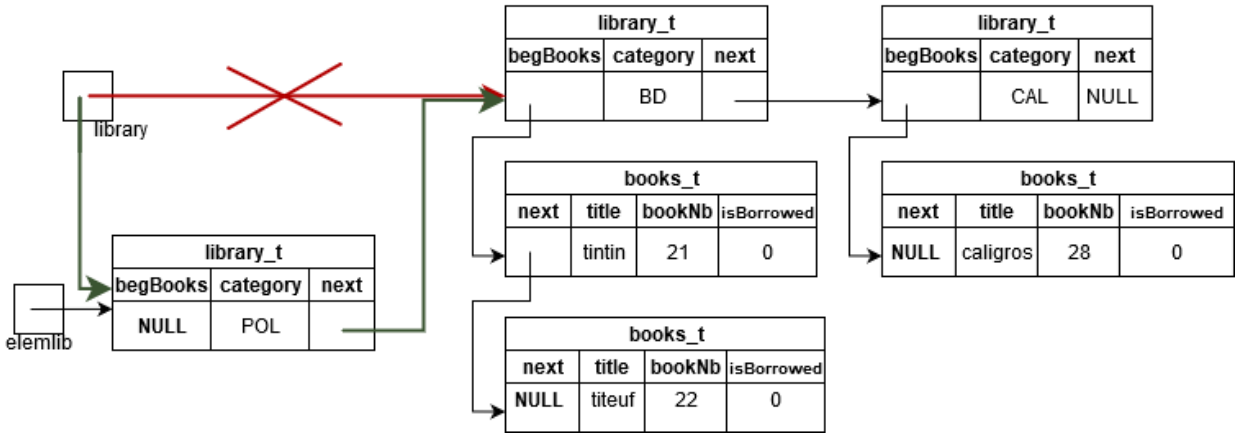
createCategory :

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

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



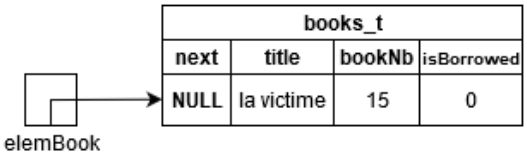
2-On insert en la nouvelle catégorie en tête de la bibliotheque



createBook :

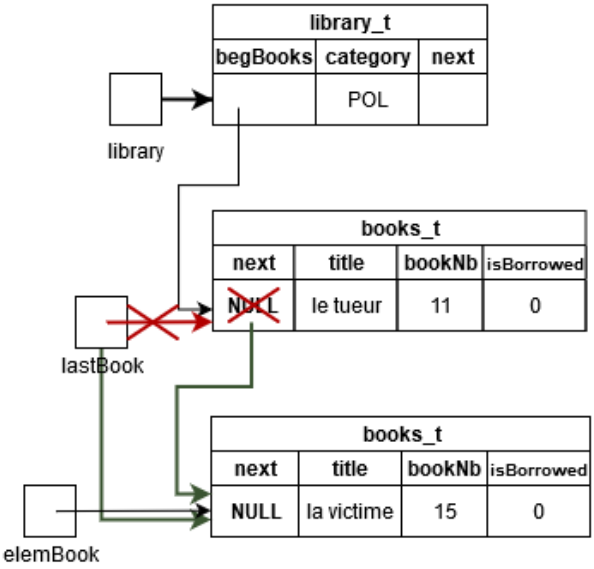
Créer un livre et l'ajoute à la fin de la liste des livres.

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

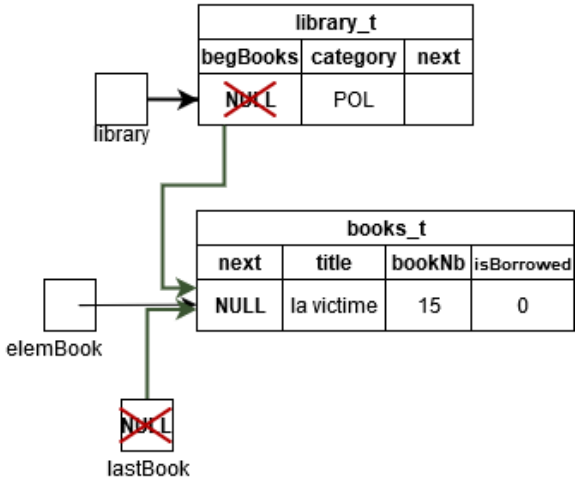


2-On ajoute le livre à la liste des livres de la catégorie (2 cas)

Si il y a déjà un livre dans la categorie



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



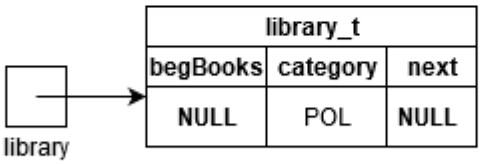
createLibrary :

Créer la bibliothèque.

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

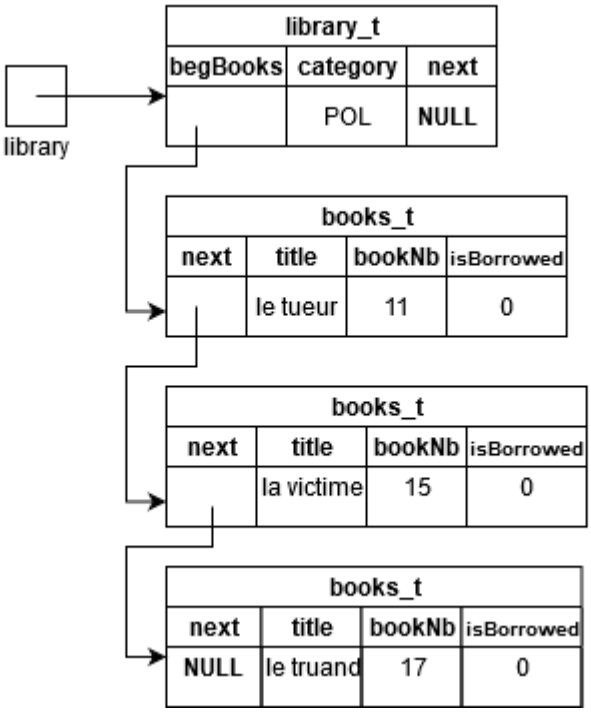
2-Tant que le fichier n'est pas lu complètement, on ajoute en tête la nouvelle catégorie

Appel procédure createCategory



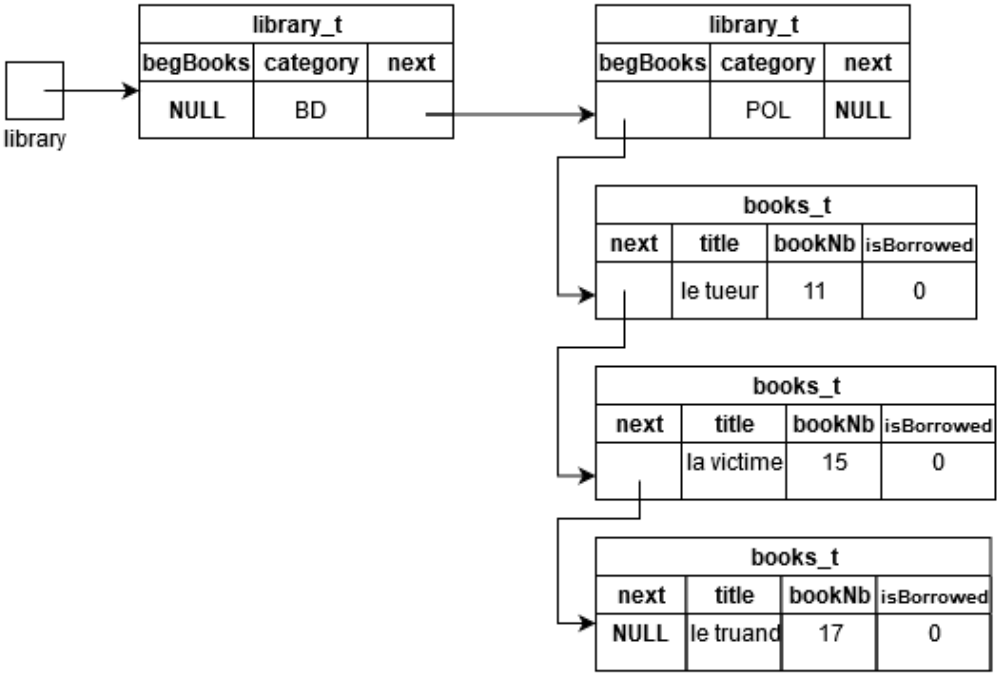
3-On ajoute les livres de la catégorie

Appel procédure createBook



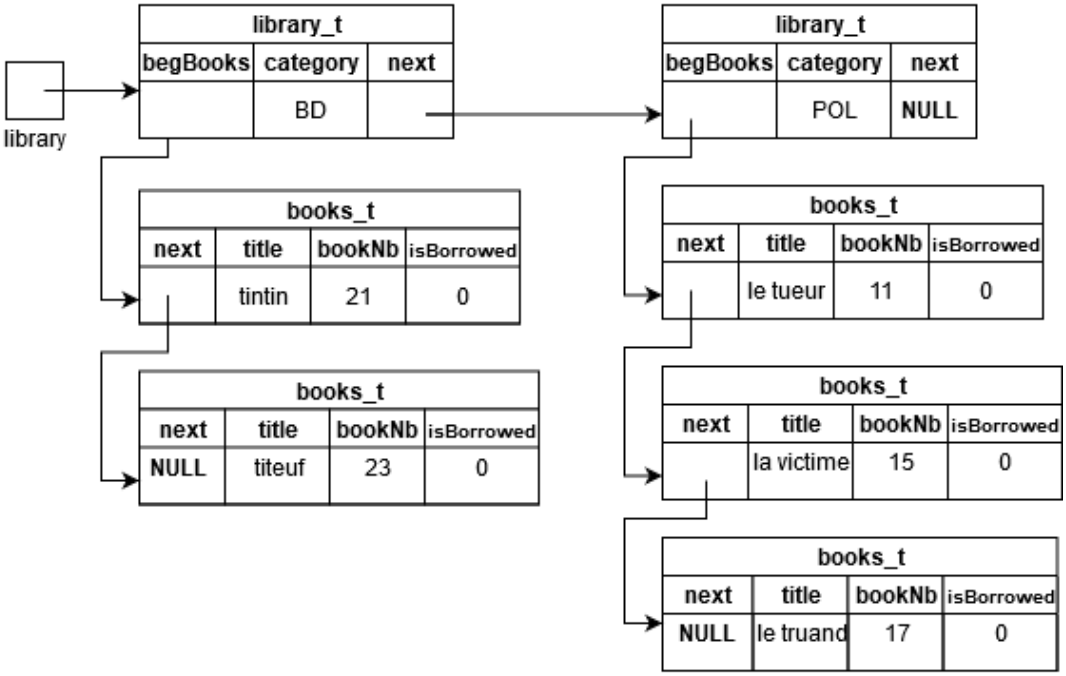
2-Tant que le fichier n'est pas vide, on ajoute en tête la nouvelle catégorie

Appel procédure createCategory



3-On ajoute les livres de la catégorie

Appel procédure createBook



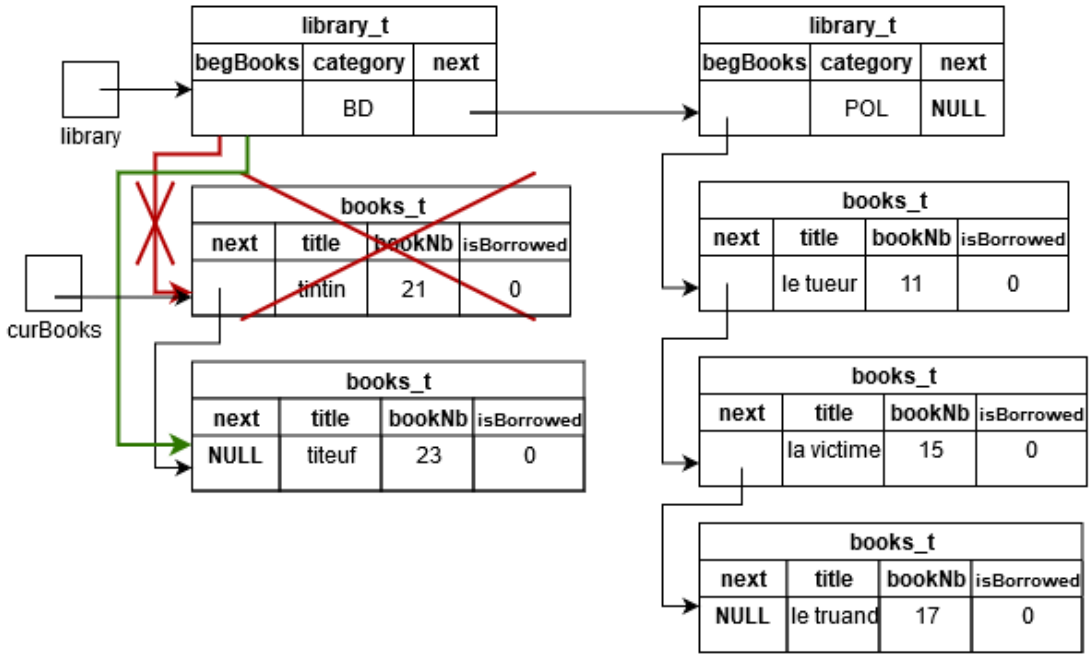
Et ainsi de suite jusqu'à avoir parcourue tout le fichier

freeLibrary :

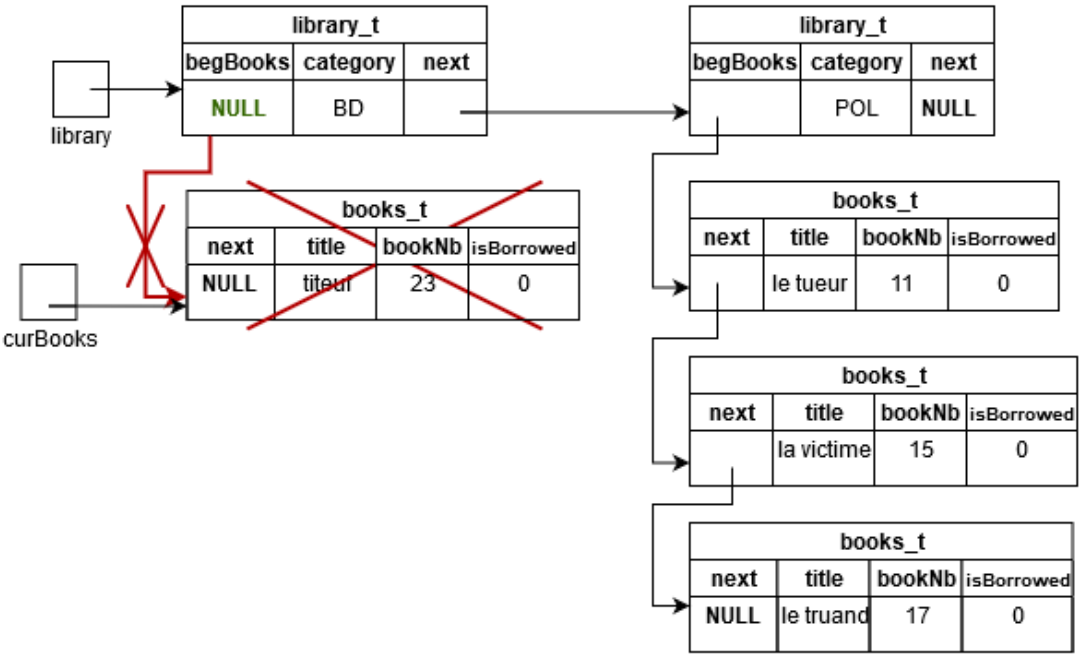
Libère la mémoire de la bibliothèque.

On supprime la première catégorie

On supprime le premier livre de cette catégorie

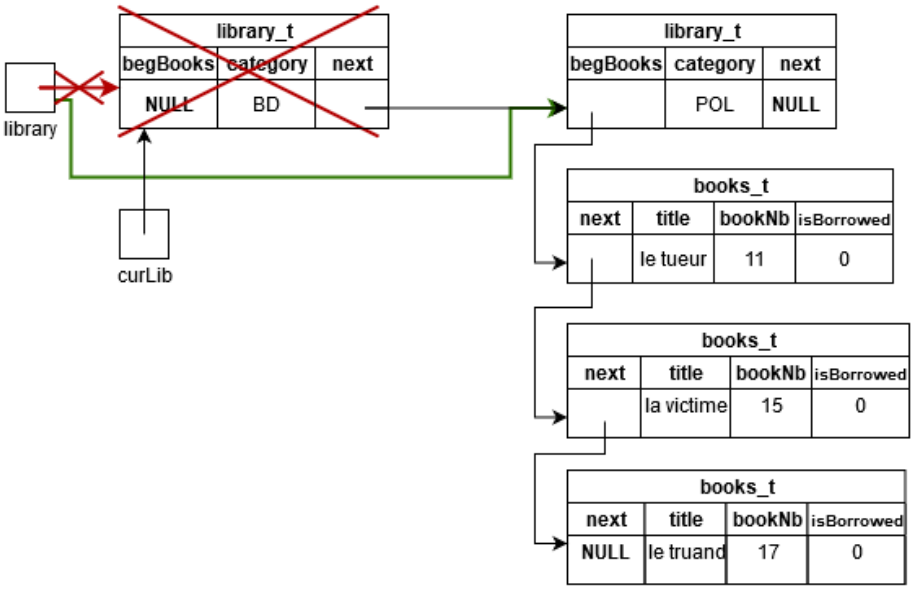


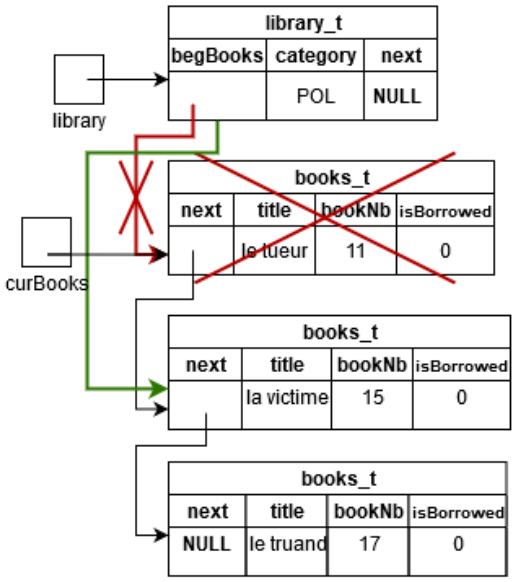
Tant qu'il y a des livres dans la catégories, on supprime celui qui est en tête

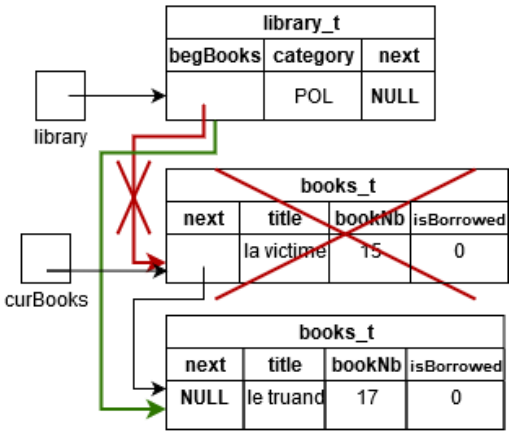


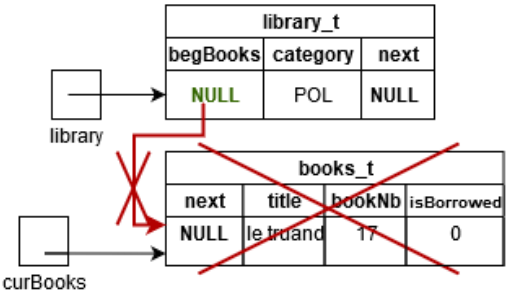
Tant qu'il y a des catégories dans la bibliothèque, on continue

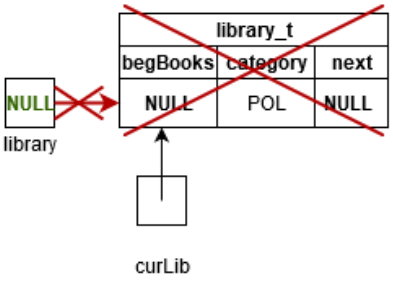
Tant qu'il y a des catégories dans la bibliothèque, on continue









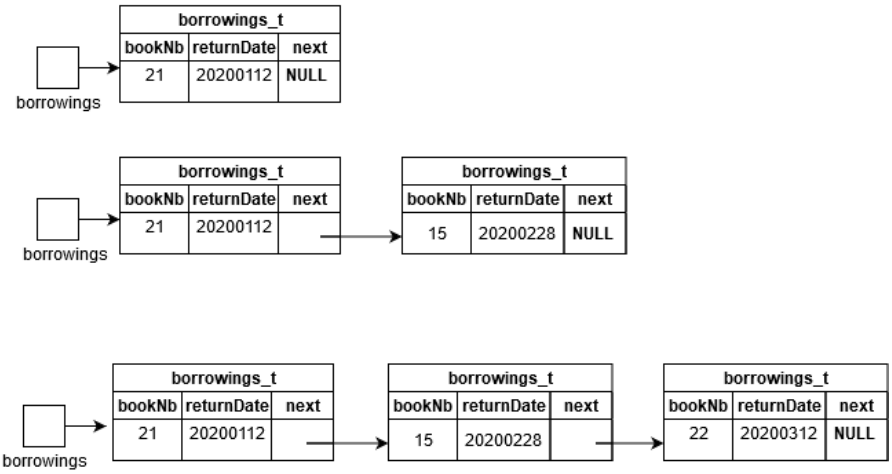


borrowBook :

Lit le fichier des emprunts et rempli la liste chainée.

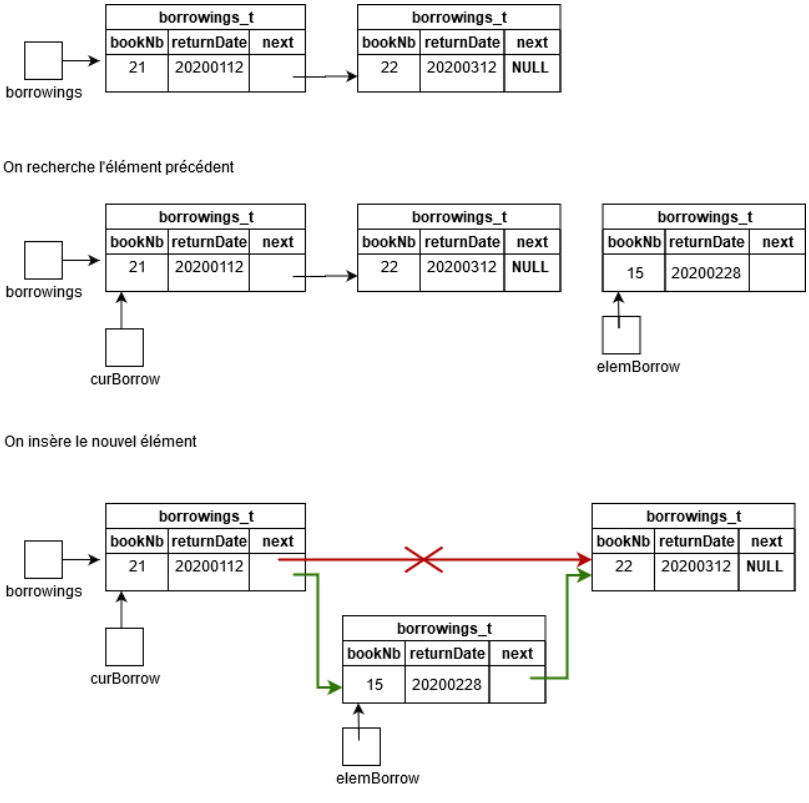
1-On part du pointeur sur la liste des emprunts

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



insertBorrowing :

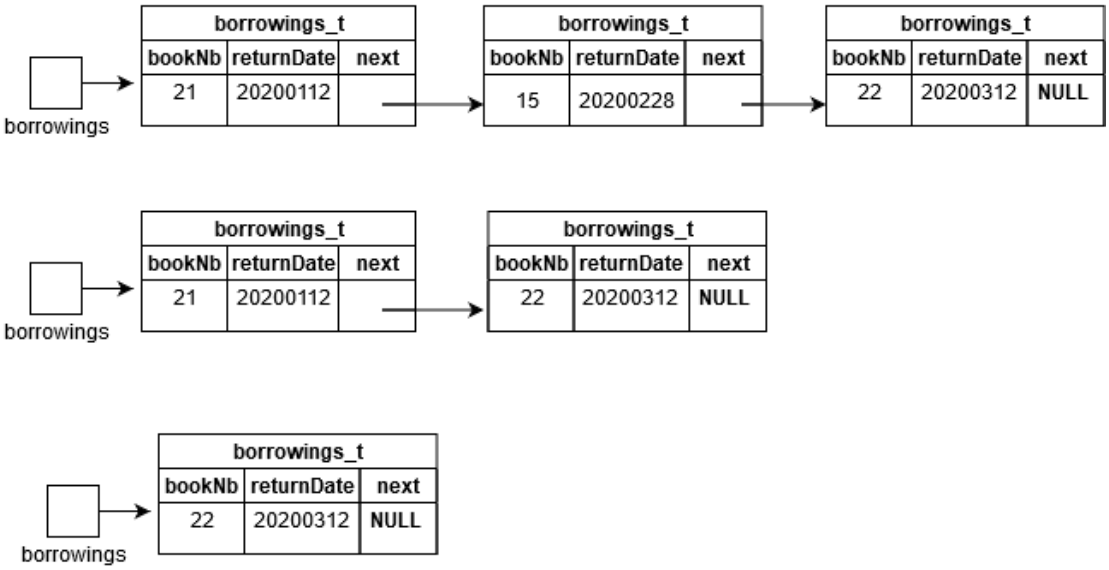
Insère le livre emprunté dans la liste des emprunts et modifie la valeur de isBorrowed du livre dans la bibliothèque.



broughtBackBook :

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

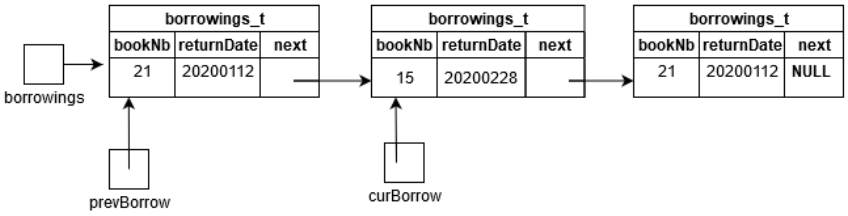
1-Tant que le ficher des retours n'est pas lu complétement,
on supprime les emprunts écrient dans le fichier
Appel de la procédure deleteBorrowing



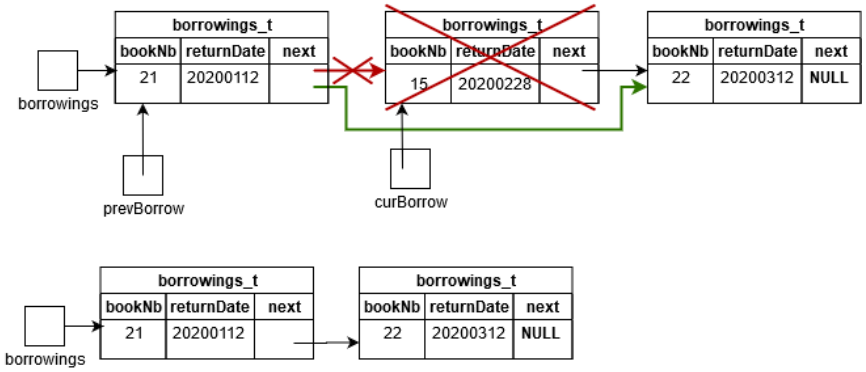
deleteBorrowing :

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

1-On cherche l'emprunt précédent celui à supprimer



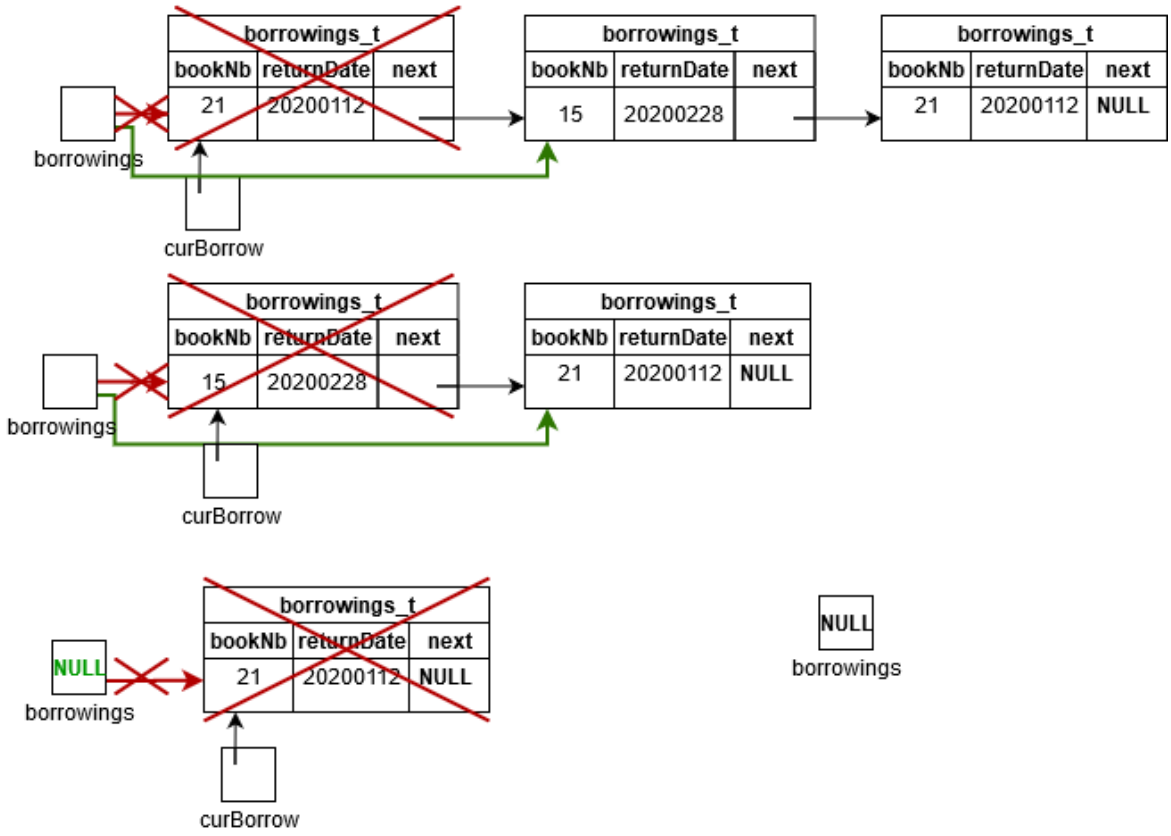
2-On supprime l'emprunt



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 a 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 a 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 a 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 a 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 Tout Rendus

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 POL 11 33333333 POL 17 22222222 BD 23 11111111	BD 21 CAL 28 BD 23 JAP 12 POL 11 POL 15 POL 17

```
| 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 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 a rendre avant le...  : 4 |
|    Sauvegarder les emprunts dans un fichier   : 5 |
|                  Quitter                     : 0 |
-----------------------------------------------------
       -: 2

  On affiche la liste des emprunts :

  Liste emprunts vide
```

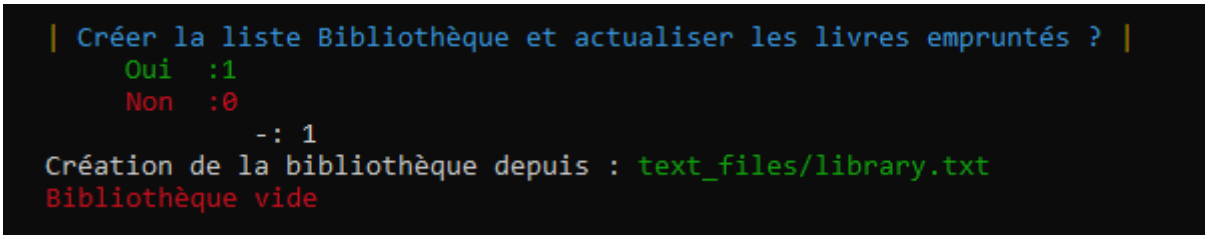
Tout Empruntés et Rien Rendus

library.txt	borrowings.txt	brought_back.txt
POL 3 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 POL 11 33333333 POL 17 22222222 BD 23 11111111	

```
| 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
```

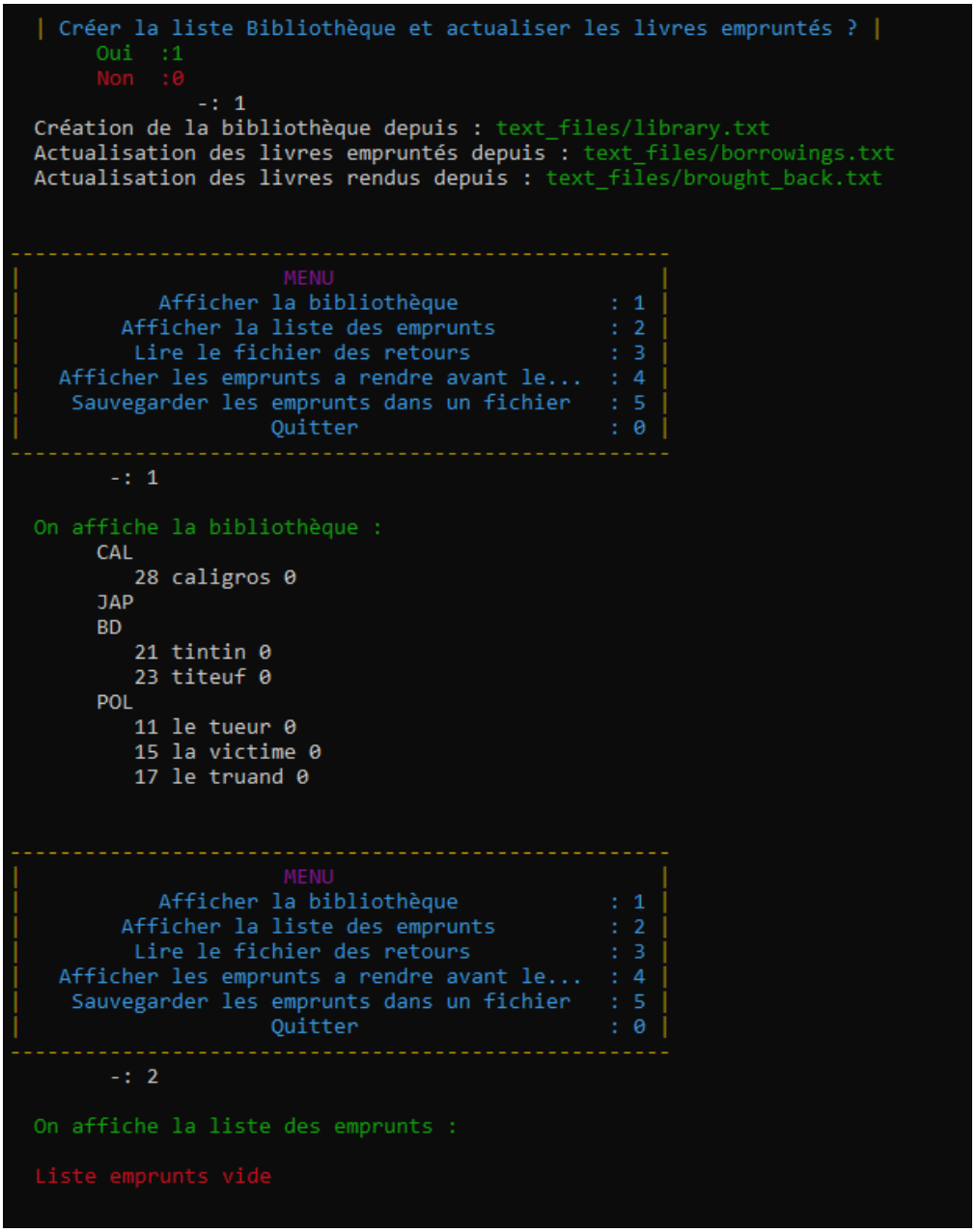
library vide

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



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



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 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 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 a 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

```
#options
CFLAGS = -Wall -Wextra -g -MMD
LIB = -lm

SRC = $(wildcard src/*.c)
OBJ = $(patsubst src/%.c,build/%.o,$(SRC))
DEP = $(patsubst %.o,%.d,$(OBJ))

.PHONY: all
all: bin/executable
    @echo "Executer avec ./bin/executable text_files/library.txt"
    @mkdir -p "borrowings"

#édition des liens
bin/executable: $(OBJ)
    @mkdir -p $(@D) #créer le dossier bin, s'il n'existe pas
    gcc -o $@ $^ $(LIB)

#génère les fichiers .o (dans build) à partir des fichiers .c (dans src) du même nom
#comppilation
build/%.o: src/%.c
    @mkdir -p $(@D) #créer le dossier build, s'il n'existe pas
    gcc -c $< $(CFLAGS) -o $@

.PHONY: clean
clean:
    rm -rf build

-include $(DEP)

```

structures

```c
/* ------------------------------------------------------------------------- */
/*            Structure pour la liste des livres d'une catégorie             */
/*                                Liste non triée                            */
/* ------------------------------------------------------------------------- */
typedef struct library {
    char category[4];
    struct books * begBooks;
    struct library * next;
}library_t;
```

```c
/* ------------------------------------------------------------------------- */
/*           Structure pour la liste des catégories (bibliothèque)           */
/*                    Triée selon l'entier bookNb croissant                  */
/* ------------------------------------------------------------------------- */
typedef struct books {
    int bookNb;
    char title[11];
    char isBorrowed;
    struct books * next;
}books_t;
```

```c
/* ------------------------------------------------------------------------- */
/*                    Structure pour la liste des emprunts                   */
/*                    Triée sur la date retour en croissant                  */
/* ------------------------------------------------------------------------- */
typedef struct borrowings {
    int bookNb;
    char returnDate[9]; // Forme AAAAMMJJ
    struct borrowings * next;
}borrowings_t;
```

main

main.c

```c
/* ------------------------------------------------------------------------ */
/*  main.c                                                                  */
/*                    Contient le programme principal                       */
/*                                                                          */
/* ------------------------------------------------------------------------ */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "common.h"
#include "borrow.h"
#include "lists.h"
#include "menu.h"


int main(int argc, char ** argv) {
    if (argc<2) {
        printf("Entrez un nom de fichier à lire\n");

    } else {
        library_t    * library    = NULL; /*Bibliothèque*/
        borrowings_t * borrowings = NULL; /*Liste emprunts*/

        int            isStarted  = 0;    /*On lance le programme ou non*/

        char           filename[22];      /*Nom du fichier à appeler*/
        char           filePath[40];      /*Chemin vers le fichier à appeler*/

        printf("\n\033[33m   | \033[36mCréer la liste Bibliothèque et actualiser les livres empruntés ?\033[33m |\n" \
               "\t\033[32mOui  :1\n\t\033[31mNon  :0\033[00m\n\t\t-: ");
        scanf("%d",&isStarted);

        /*Le programme se lance*/
        if (isStarted == 1) {
            /*Création de la bibliothèque depuis le fichier passé en paramètre lors de l'execution*/
            createLibrary(argv[1], &library);

            printf("   Création de la bibliothèque depuis : \033[32m%s\033[00m\n", argv[1]);

            if (library != NULL) {
                /*Cherche s'il existe un fichier des emprunts dans le dossier borrowings*/
                /*et sélectionne le plus récent*/
                findFilenameMax(filename);

                /*Créer le bon filePath à partir de filename*/
                if (strcmp(filename, "0000-00-00_00h00'00''")) {
                    snprintf(filePath, 40, "./borrowings/%s", filename);
                } else {
                    strcpy(filePath, "text_files/borrowings.txt");
                }

                    /*Création de la liste des emprunts à partir de filePath*/
                    borrowBook(filePath, library, &borrowings);
                    printf("   Actualisation des livres empruntés depuis : \033[32m%s\033[00m\n", filePath);

                    strcpy(filePath, "text_files/brought_back.txt");
                    /*Actualise les emprunts en fonction des retours à partir de filePath*/
                    broughtBackBook(filePath, library, &borrowings);
                    printf("   Actualisation des livres rendus depuis : \033[32m%s\033[00m\n", filePath);

                    /*Lancement du menu*/
                    menu(&library, &borrowings);

                } else {
                    printf("\033[31m   Bibliothèque vide\033[00m\n\n");
                }

                /*Libération de la mémoire*/
                freeLibrary(&library);
                freeBorrowings(&borrowings);

            } else {
                printf("\033[31m   Quitter\n\n\033[00m");
            }
        }
        return 0;
    }
```

```c
/* ------------------------------------------------------------------------ */
/* menu         Fonction de menu (graphique)                                */
/*              On reste dans cette fonction tant qu'on ne choisi pas 0     */
/*                                                                          */
/* En entrée:   library    : pointeur sur la bibliothèque (par adresse)     */
/*              borrowings : pointeur sur la liste emprunts (par adresse)   */
/*                                                                          */
/* En sortie:   void                                                        */
/* ------------------------------------------------------------------------ */
void menu(library_t **, borrowings_t **);
```

```c
void menu(library_t ** library, borrowings_t ** borrowings) {
    int  choice = -1;   /*Choix dans le menu*/
    int  argNb  = 0;    /*Evite les bugs liés à une saisie incorecte pour le choix du menu*/
    char filename[255]; /*Pour stocker le nom du fichier à ouvrir*/

    while (choice != 0) {

        printf("\033[33m\n\n" \
            " ------------------------------------------------------\n" \
            " |\033[35m                      MENU                            \033[33m|\n" \
            " |\033[36m           Afficher la bibliothèque            : 1 \033[33m|\n" \
            " |\033[36m         Afficher la liste des emprunts        : 2 \033[33m|\n" \
            " |\033[36m          Lire le fichier des retours          : 3 \033[33m|\n" \
            " |\033[36m   Afficher les emprunts a rendre avant le...  : 4 \033[33m|\n" \
            " |\033[36m    Sauvegarder les emprunts dans un fichier   : 5 \033[33m|\n" \
            " |\033[36m                    Quitter                    : 0 \033[33m|\n" \
            " ------------------------------------------------------\n\033[00m" \
            "         -: ");

        // Dans un scanf : %n permet de récupérer le nombre de caractères lu par le scanf
        // scanf return le nombre d'arguments qu'elle a rempli (l'argument %n n'est pas pris en compte)
        argNb = scanf("%n%d", &argNb, &choice);
        // On vide le buffer dans le cas où autre chose qu'un int a été entré
        while (getchar () != '\n');

        if (argNb == 1) {

            switch (choice){

                case 0:
                    printf("\n\033[31m   On quitte\033[00m\n\n");
                    break;

                case 1:
                    displayLibrary(*library);
                    break;

                case 2:
                    displayBorrowings(*borrowings);
                    break;

                case 3:
                    menuChoice3(*library, borrowings);
                    break;

                case 4:
                    menuChoice4(*borrowings);
                    break;

                case 5:
                    saveBorrowingsInFile(filename, *library, *borrowings);
                    break;

                default:
                    printf("\n\033[33m   Selectionner une option\033[00m\n");
                    break;
            }

        } else {
            printf("\n\033[31m   Choix Invalide\033[00m\n");
        }
    }
}
```

```c
/* ------------------------------------------------------------------------ */
/* menuChoice3      Action à executer lors du choix 3 dans le menu          */
/*                                                                          */
/* En entrée:  library    : pointeur sur la bibliothèque (par valeur)       */
/*             borrowings : pointeur sur la liste emprunts (par adresse)    */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void menuChoice3(library_t *, borrowings_t **);
```

```c
void menuChoice3(library_t * library, borrowings_t ** borrowings) {
    char filename[255];     /*Pour stocker le nom du fichier à ouvrir*/
    int  typeFilename = 1;  /*Booléen -saisie du nom du fichier ou -nom automatique*/
    char lineTmp[255];      /*Chaine de caractères pour récupérer la saisie avec fgets()*/

    printf("\n" \
           "\033[32m   |\033[36m Nom du fichier des rendus :          \033[32m|\n" \
                   "   |\033[36m    text_files/brought_back.txt   : 1 \033[32m|\n" \
                   "   |\033[36m    Autre nom à entrer            : 0 \033[32m|\033[00m\n" \
           "         -: ");
    fgets(lineTmp, 255, stdin);
    typeFilename = lineTmp[0] - 48;

    if (typeFilename == 0) {
        printf("\n\033[32m   |\033[36m Entrer le nom du fichier des retours : \033[32m|\033[00m\n" \
                "         -: ");
        fgets(filename, 255, stdin);
        remove_endstr_n(filename);

        printf("   Actualisation des livres empruntés depuis le fichier des retours :\n" \
               "         \033[32m%s\033[00m\n", filename);
        broughtBackBook(filename, library, borrowings);

    } else if (typeFilename == 1) {
        strcpy(filename, "text_files/brought_back.txt");
        printf("   Actualisation des livres empruntés depuis le fichier des retours :\n" \
               "         \033[32m%s\033[00m\n", filename);
        broughtBackBook(filename, library, borrowings);

    } else {
        printf("\n\033[31m   Choix Invalide\033[00m\n");
    }
}
```

```c
/* ------------------------------------------------------------------------ */
/* menuChoice4      Action à executer lors du choix 4 dans le menu          */
/*                                                                          */
/* En entrée:  borrowings : pointeur sur la liste emprunts (par valeur)     */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void menuChoice4(borrowings_t *);
```

```c
void menuChoice4(borrowings_t * borrowings) {
    char date[255]; /*date saisie*/

    printf("\n" \
           "\033[32m   |\033[36m Selectionner une date AAAAMMJJ : \033[32m|\n" \
           "\033[00m         -: ");

    fgets(date, 255, stdin);
    remove_endstr_n(date);

    if (isDateInputCorrect(date)) {
        displayBorrowingsBeforeDate(borrowings, date);

    } else {
        printf("\n\033[31m   Date Invalide\033[00m\n");
    }
}
```

```c
/* ------------------------------------------------------------------------ */
/* remove_endstr_n        Supprime le caractère \n de fin de chaine         */
/*                        si il existe                                      */
/*                                                                          */
/* En entrée:  line : chaine de caractères                                  */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void remove_endstr_n(char * line);
```

```c
void remove_endstr_n(char * line){
    int i = 0; /*Compteur*/

    while (line[i]!='\0') {
        i++;
    }
    i--;

    if (line[i] == '\n') {
        line[i] = '\0';
    }
}
```

```
65  /* ------------------------------------------------------------------------ */
66  /* isDateInputCorrect    Verifie si la chaine passée en paramètre est       */
67  /*                       une date de la forme AAAAMMJJ                      */
68  /*                                                                          */
69  /* En entrée:  date : chaine de caractères                                  */
70  /*                                                                          */
71  /* En sortie:  int  : 1 si reussi (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
```

```c
/* ------------------------------------------------------------------------ */
/* findFileName   Modifie la chaine de caractères passées en paramètre,     */
/*                Celle ci prend le  nom du dernier fichier emprunts        */
/*                créé, à lire lors de la prochaine excecution du programme */
/*                                                                          */
/* En entrée:  filenameMax : chaine de caractères                           */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void findFilenameMax(char [22]);
```

```c
void findFilenameMax(char filenameMax[22]) {
    DIR * rep = opendir("./borrowings");   /*Pointeur répertoire*/
    strcpy(filenameMax, "0000-00-00_00h00'00''");

    if (rep != NULL) {
        struct dirent * ent = NULL;        /*Pointeur entitée*/
        char            filename[22];      /*Nom du fichier*/
        char            filenameDate[15];  /*Date de la forme AAAAMMJJhhmmss contenue dans le nom du fichier*/
        char            filenameDateMax[15] = "00000000000000";
        int             i = 0;
        int             j = 0;

        while ((ent = readdir(rep)) != NULL) {

            if (strcmp(ent->d_name, ".") && strcmp(ent->d_name, "..")) {

                strcpy(filename, ent->d_name);

                /*Remplissage de filenameDate avec uniquement les chiffres contenus dans filename*/
                i = 0; j = 0;
                while (filename[i] != '\0') {
                    if (filename[i] >= 48 && filename[i] <= 57) {
                        filenameDate[j] = filename[i];
                        j++;
                    }
                    i++;
                }
                filenameDate[14] = '\0';

                /*filenameMax contient le nom du fichier le plus récent s'il existe, "0000-00-00_00h00'00'" sinon*/
                if (atoi(filenameDate) > atoi(filenameDateMax)) {
                    strcpy(filenameMax, filename);
                }
            }

        }

        closedir(rep);

    }
}
```

List

```c
/* ------------------------------------------------------------------------ */
/* createLibrary   Creer la bibliothèque (liste chainée library)            */
/*                                                                          */
/* En entrée:  filename : nom du fichier à lire pour remplir la liste       */
/*             library  : pointeur sur la bibliothèque (par adresse)        */
/*                                                                          */
/* En sortie:  int : Retour d'erreur, 1 si reussi, 0 sinon                  */
/* ------------------------------------------------------------------------ */
int createLibrary(char *, library_t **);
```

```c
int createLibrary(char * filename, library_t ** library) {
    FILE * file = NULL;          /*Fichier*/
    file = fopen(filename, "r");

    int       error = 1;         /*Retour d'erreur*/

    char      category[4]  = "_"; /*Catégorie lue dans le fichier*/
    int       categorySize = 0;   /*Nombre de livre dans une catégorie*/

    int       i = 0;             /*Compteur*/
    books_t * lastBook = NULL;   /*Dernier livre creé*/

    int       bookNb = 0;        /*Numéro du livre lu dans le fichier*/
    char      title[11];         /*Titre lu dans le fichier*/

    if (file != NULL) {

        while (!feof(file) && error == 1) {

            fscanf(file, "%s %d", category, &categorySize);

            // verification importante dans le cas où le fichier ne contient qu'une ligne avec \n
            if (category[0] != '_') {
                error = createCategory(library, category);

                if (error == 1){

                    for (i = 0; i < categorySize; i++) {
                        // %[^\n] pour récupérer les chaines de caracteres comportants des espaces jusqu'au premier \n à l'aide d'un scanf
                        fscanf(file, "%d %[^\n]", &bookNb, title);
                        remove_endstr_r_windows(title);
                        error = createBook(library, &lastBook, bookNb, title);
                    }
                }
            }
        }
        fclose(file);

    } else {
        error = 0;
    }
    return error;
}
```

```c
/* ------------------------------------------------------------------------ */
/* createCategory   Creer une categorie et l'ajoute dans la bibliothèque (en tête) */
/*                                                                          */
/* En entrée:  library  : pointeur sur la bibliothèque (par adresse)        */
/*             category : nom de la catégorie en 3 caractères max           */
/*                                                                          */
/* En sortie:  int : Retour d'erreur, 1 si reussi, 0 sinon                  */
/* ------------------------------------------------------------------------ */
int createCategory (library_t **, char[4]);
```

```c
int createCategory(library_t ** library, char category[4]) {
    int          error = 1;    /*Retour d'erreur*/
    library_t * elemLib = (library_t *)malloc(sizeof(library_t));

    if (elemLib != NULL) {
        strcpy(elemLib->category, category);
        elemLib->begBooks = NULL;
        elemLib->next = *library;
        *library = elemLib;

    } else {
        error = 0;
    }

    return error;
}
```

```
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
95      } else {
96          error = 0;
97      }
98      return error;
99  }
```

```c
/* ------------------------------------------------------------------------ */
/* displayLibrary      Affiche la bibliothèque                              */
/*                                                                          */
/* En entrée:  curLib : pointeur sur la bibliothèque (par valeur)           */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void displayLibrary(library_t const *);
```

```c
void displayLibrary(library_t const * curLib) {
    books_t * curBooks = NULL; /*Pointeur courant sur la liste des livres d'une catégorie*/

    printf("\n\033[32m   On affiche la bibliothèque :\033[00m\n");

    if (curLib != NULL) {
        while (curLib != NULL) {
            printf("\t%s\n", curLib->category);
            curBooks = curLib->begBooks;

            while (curBooks != NULL) {
                printf("\t   %d %s %d\n", curBooks->bookNb, curBooks->title, curBooks->isBorrowed);
                curBooks = curBooks->next;
            }

            curLib = curLib->next;
        }

    } else {
        printf("\n\033[31m   Liste bibliothèque vide\033[00m\n");
    }
}
```

```c
/* ------------------------------------------------------------------------ */
/* remove_endstr_r_windows   Supprime le caractère \r de fin de chaine      */
/*                           si il existe                                   */
/*                                                                          */
/* En entrée:  line : chaine de caractères                                  */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void remove_endstr_r_windows(char *);
```

```c
void remove_endstr_r_windows(char * line){
    int i = 0; /*Compteur*/

    while (line[i]!='\0') {
        i++;
    }
    i--;

    if (line[i] == '\r') {
        line[i] = '\0';
    }
}
```

```c
/* ------------------------------------------------------------------------ */
/* freeLibrary            Libère la mémoire                                 */
/*                                                                          */
/* En entrée:  library : pointeur sur la bibliothèque (par adresse)         */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void freeLibrary(library_t **);
```

```c
void freeLibrary(library_t ** library) {
    library_t * curLib   = *library;  /*Pointeur courant sur la bibliothèque*/
    books_t   * curBooks = NULL;      /*Pointeur courant sur la liste des livres d'une catégorie*/

    /*Libération de la bibliothèque*/
    while (*library != NULL) {
        curBooks = curLib->begBooks;

        /*Libération des livres d'une catégorie*/
        while (curLib->begBooks != NULL) {
            curBooks = curBooks->next;
            free(curLib->begBooks);
            curLib->begBooks = curBooks;
        }

        /*Libération des catégories*/
        curLib = curLib->next;
        free(*library);
        *library = curLib;
    }
}
```

Borrow

```c
/* ------------------------------------------------------------------------ */
/* borrowBook  Lit le fichier des emprunts et rempli la liste chainée       */
/*                                                                          */
/* En entrée:  filename   : chaine de caractères, nom du fichier            */
/*             library    : pointeur sur la bibliothèque (par valeur)       */
/*             borrowings : pointeur sur la liste emprunts (par adresse)    */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void borrowBook(char *, library_t *, borrowings_t **);
```

```c
void borrowBook(char * filename, library_t * library, borrowings_t ** borrowings) {
    FILE * file = NULL;   /*Fichier*/
    file = fopen(filename, "r");

    books_t * bookBorrowed = NULL;

    if (library != NULL) {

        if (file != NULL) {
            char category[4] = "_"; /*Categorie lue dans le fichier*/
            int  bookNb      = 0;   /*Numero du livre lu dans le fichier*/
            char date[9];           /*Date de rendu lue dans le fichier*/

            while (!feof(file)) {
                fscanf(file, "%s %d %s", category, &bookNb, date);

                // Verification importante dans le cas où le fichier ne contient qu'une ligne avec \n
                if (category[0] != '_') {
                    bookBorrowed = isBookInLibrary(library, category, bookNb); /*bookBorrowed == NULL si le livre n'a pas été trouvé dans la bibliothèque*/
                    /*On insère le livre dans la liste des emprunts*/
                    insertBorrowing(borrowings, bookBorrowed ,date);
                }
            }
            fclose(file);

        } else {
            printf("\n\033[31m   Nom de fichier pour les emprunts inexistant\033[00m\n");
        }

    } else {
        printf("\n\033[31m   Bibliotheque vide\033[00m\n");
    }
}
```

```
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 : categorie 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
82      return curBooks;
83  }
```

```c
/* ------------------------------------------------------------------------ */
/* insertBorrowing    Insere le livre emprunté dans la liste emprunts et    */
/*                    modifie la valeur de isBorrowed dans la bibliothèque  */
/*                                                                          */
/* En entrée:  borrowings   : pointeur sur la liste emprunts (par adresse)  */
/*             bookBorrowed : adresse du livre de la bibliothèque que l'on  */
/*                            souhaite emprunter                            */
/*             date         : chaine de caractère, date retour du livre     */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void insertBorrowing(borrowings_t **, books_t * curBooks, char[9]);
```

```c
void insertBorrowing(borrowings_t ** borrowings, books_t * bookBorrowed, char date[9]) {
    borrowings_t * curBorrow  = *borrowings; /*Pointeur courant sur la liste des emprunts*/
    borrowings_t * elemBorrow = NULL;        /*Element alloué*/

    /*Modifie la valeur de isBorrowed dans la bibliothèque*/
    if (bookBorrowed != NULL && bookBorrowed->isBorrowed != 1) {
        bookBorrowed->isBorrowed = 1;

        elemBorrow = (borrowings_t *)malloc(sizeof(borrowings_t));

        if (elemBorrow != NULL) {
            elemBorrow->bookNb = bookBorrowed->bookNb;
            strcpy(elemBorrow->returnDate, date);

            /*Place le livre au bon endroit dans la bibliothèque (trié par date croissante)*/
            if (curBorrow != NULL && atoi(date) > atoi(curBorrow->returnDate)) {
                while (curBorrow->next != NULL && atoi(date) > atoi(curBorrow->next->returnDate)) {
                    curBorrow=curBorrow->next;
                }
                elemBorrow->next = curBorrow->next;
                curBorrow->next = elemBorrow;
            } else {
                *borrowings = elemBorrow;
                elemBorrow->next = curBorrow;
            }
        }

    /*else : Livre de la liste à emprunter inexistant*/
    }
}
```

```c
/* ------------------------------------------------------------------------ */
/* broughtBackBook  Lit le fichier des retours, supprime les livres         */
/*                  rendus de la liste des emprunts et modifie la           */
/*                  valeur de isBorrowed dans la bibliothèque               */
/*                                                                          */
/* En entrée:  filename   : chaine de caractère, nom du fichier des retours*/
/*             library    : pointeur sur la bibliothèque (par valeur)      */
/*             borrowings : pointeur sur la liste emprunts (par adresse)   */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void broughtBackBook(char *, library_t *, borrowings_t **);
```

```c
void broughtBackBook(char * filename, library_t * library, borrowings_t ** borrowings) {
    FILE * file = NULL;   /*Fichier*/
    file = fopen(filename, "r");

    if (library != NULL) {

        if (file != NULL) {
            char category[4] = "_"; /*Categorie lue dans le fichier*/
            int  bookNb      = 0;   /*Numero du livre lu dans le fichier*/

            while (!feof(file)) {
                fscanf(file, "%s %d", category, &bookNb);

                // Verification importante dans le cas où le fichier ne contient qu'une ligne avec \n
                if (category[0] != '_') {
                    /*Supprime le livre de la liste emprunts*/
                    deleteBorrowing(borrowings, bookNb);
                    /*Modifie la valeur de isBorrowed dans la bibliothèque*/
                    isBorrowedToFalse(library, category, bookNb);
                }
            }

            fclose(file);
        } else {
            printf("\n\033[31m   Nom de fichier pour les retours inexistant\033[00m\n");
        }

    } else {
        printf("\n\033[31m   Liste bibliotheque vide\033[00m\n");
    }
}
```

```c
/* ------------------------------------------------------------------------ */
/* deleteBorrowing   Supprime de la liste des emprunts le livre dont        */
/*                   le numéro est passé en paramètre (libération mémoire)*/
/*                                                                          */
/* En entrée:  borrowings : pointeur sur la liste emprunts (par adresse)    */
/*             bookNb     : entier, numéro du livre à supprimer             */
/*                                                                          */
/* En sortie:  void                                                         */
/* ------------------------------------------------------------------------ */
void deleteBorrowing(borrowings_t ** , int);
```

```c
void deleteBorrowing(borrowings_t ** borrowings, int bookNb) {
    borrowings_t * curBorrow  = *borrowings; /*Pointeur courant sur la liste des emprunts*/
    borrowings_t * prevBorrow = *borrowings; /*Pointeur courant précédent sur la liste des emprunts*/

    if (*borrowings != NULL) {

        while (curBorrow != NULL && curBorrow->bookNb != bookNb) {
            prevBorrow = curBorrow;
            curBorrow = curBorrow->next;
        }

        if (curBorrow != NULL) {
            prevBorrow->next = curBorrow->next;
        } else {
            prevBorrow->next = NULL;
        }

        if (curBorrow == *borrowings) {
            *borrowings = (*borrowings)->next;
        }

        free(curBorrow);
    }
}
```

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

```
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 }
```

```
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      : chaine 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  }
```

```
115  /* ------------------------------------------------------------------------ */
116  /* createFilename  Remplie la chaine de caractères passée en paramètre      */
117  /*                 avec la date et l'heure à laquelle est créé le fichier   */
118  /*                                                                          */
119  /* En entrée:  filename : chaine 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      snprintf(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  : chaine 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
247          fclose(file);
248
249      }
250  }
```

```c
/* ------------------------------------------------------------------------ */
/* findCategoryName   Remplie la chaine de caractères passée en paramètre */
/*                    avec le nom de la categorie correspondant au numéro */
/*                    du livre                                            */
/*                                                                        */
/* En entrée:  curLib   : pointeur courant sur la liste bibliothèque      */
/*             bookNb   : entier, numéro du livre                         */
/*             category : chaine de caractères à remplir                  */
/*                                                                        */
/* En sortie:  void                                                       */
/* ------------------------------------------------------------------------ */
void findCategoryName(library_t const *, int, char[4]);
```

```c
void findCategoryName(library_t const * curLib, int bookNb, char category[4]) {
    int       isfound  = 0;    /*Livre trouvé (1) ou non (0)*/
    books_t * curBooks = NULL; /*Pointeur courant sur la liste des livres d'une catégorie*/

    while (curLib != NULL && isfound == 0) {
        strcpy(category, curLib->category);
        curBooks = curLib->begBooks;

        while (curBooks != NULL && curBooks->bookNb != bookNb) {
            curBooks = curBooks->next;
        }

        if (curBooks != NULL) {
            isfound = 1;
        }

        curLib = curLib->next;
    }
}
```

```
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
```