École Nationale Supérieure d'Informatique et d'Analyse des Systèmes - RABAT

Rapport de Projet de programmation : GESTION DE PRÊTS DE LIVRES

Réalisé par :

Oussama Alami
Abdelwahhab El Jabbouri

Encadré par :

Pr. Karima Moumane

Année académique 2020/2021

Remerciements :

C'est parce que nous avons beaucoup estimé tous ceux qui nous ont écoutés, conseillés, critiqués et encadrés que nous tenons à leur faire part de toutes nos gratitudes, et plus particulièrement, nous tenons à remercier à travers ces courtes lignes :

Madame Karima MOUMANE professeur à l'**ENSIAS**, qui a acceptée d'examiner et d'encadrer ce travail pour le valoriser malgré ses nombreuses occupations. Nos remerciements vont également à nos professeurs des modules de « *Structures de données* » : **Monsieur Abdellatif EL FAKER** pour son entière disponibilité, « *Algorithmique* » : **Monsieur Ahmed ETTALBI** pour le savoir qu'il nous a transmis et de « *Techniques de programmation* » : **Monsieur Hatim Guermah**. Enfin, nos remerciements s'adressent aux professeurs et au corps administratif de l'**Ecole Nationale Supérieure de l'Informatique et d'Analyse des Systèmes**.

Table des matières

Table des figures

Introduction générale

Dernièrement, beaucoup de gens reviennent aux outils informatiques afin de faciliter leurs vies quotidiennes. Ces outils peuvent être utilisés dans tous les secteurs, à savoir : le secteur professionnel, **secteur éducatif** et d'autres secteurs.

La plupart des bibliothèques ont du mal à **gérer leurs prêts de livres** de manière traditionnelle. Le rôle de l'informatique est de résoudre ce problème en développant une application de **gestion de prêts de livres** de bibliothèque.

C'est pour cela après avoir étudié plusieurs matières de « *langage C* » et de « *Structures de données* » durant le premier semestre, et afin de nous permettre de mettre en pratique les différentes notions acquises durant les cours, les TDs et les TPs, nous étions invités à développer une application en C pour **la gestion de prêts de livres** à la bibliothèque **ENSIAS**.

Le présent rapport retrace, à travers les différentes étapes que nous avons suivies pour atteindre cet objectif, ces étapes peuvent être résumées en deux chapitres :

⇒ *Le premier chapitre est consacré à la présentation de la problématique et le cadre générale du projet, ainsi que les étapes de la résolution du problème.*
⇒ *Le dernier chapitre aborde la phase développement et mise en oeuvre, avec une description des outils techniques, ainsi qu'une présentation de l'application.*

Chapitre 1

Analyse et conception

Introduction

La présente partie nous permet de présenter la problématique, identifier toutes les fonctionnalités de notre application, et donner les étapes de la résolution.

1.1 Problématique

Il s'agit de de réaliser en langage C une application permettant la gestion de prêts de livres à la bibliothèque de l'*École national supérieur d'informatique et analyse des systèmes* (**ENSIAS**).

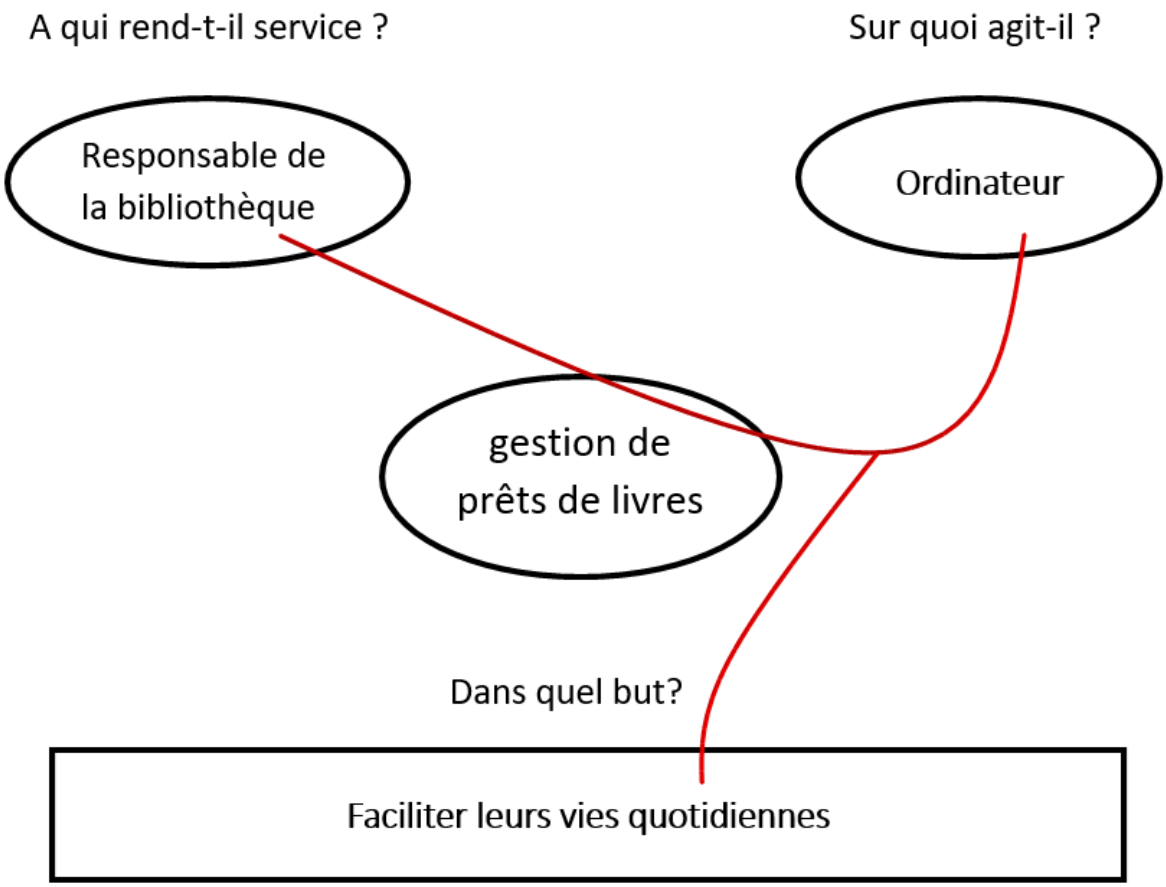

FIGURE 1.1 – Diagramme bête à corne du programme

1.2 Cahier des charges

1.2.1 Description du projet

Il s'agit de développer une application en langage C pour la gestion de prêts de livres à la bibliothèque ENSIAS. Pour le faire, nous avons besoin d'utilisé des structures de différentes natures, dont les champs contiennent les informations proposées ci-dessous :

1. **Un adhérant est caractérisé par :**
   - son numéro « *num_adh* » qui identifie **de manière unique** un adhérent
   - son nom « *nom_adh* »
   - son prénom « *prenom_adh* »
   - son adresse email « *email_adh* »
   - son adresse personnelle « *adress_adh* »
   - le nombre de livres empruntés « *nbre_emprunts_adh* ». Un adhérent peut emprunter **au maximum 3 livres**

2. **Un livre est caractérisé par :**
   - un numéro « *num_liv* » qui identifie **de manière unique** un livre
   - un titre « *titre_liv* »
   - catégorie du livre « *categ_liv* »
   - un auteur « *auteur_liv* », contenant les champs ci-dessous :
     - un nom « *nom_aut* »
     - un prénom « *prenom_aut* »
   - un numéro qui comprend le numéro de l'adhérent ayant emprunté ce livre « *emprunteur_liv* »

⇒ **Il est souhaité de créer une structure pour les informations concernant un adhérent ainsi qu'une structure pour représenter les livres.**

⇒ **Les données de ces structures sont à enregistrer dans deux fichiers.**

1.2.2 Fonctionnalités à implémenter

Le programme devra proposer à l'utilisateur un menu pour pouvoir naviguer entre les différentes fonctionnalités réalisées. En particulier la saisie des informations relatives aux adhérents et aux livres :

1. **Gestion des adhérents :**
   - Ajouter, Modifier, Supprimer un adhérent
   - Recherche d'un adhérent par son nom et affichage du résultat
   - Retour au menu principal
2. **Gestion des Livres :**
   - Ajouter, Modifier, Supprimer un livre
   - Ordonner les livres par Catégorie
   - Recherche d'un livre par Catégorie et titre du livre
   - Retour au menu principal
3. **Gestion des emprunts :**
   - Emprunter un livre
   - Afficher liste des livres empruntés
   - Afficher la liste des adhérents emprunteurs de livre
   - Rendre un livre
   - Retour au menu principal

1.3 Les étapes de la résolution du problème

Avant tout, la gestion du programme s'articule essentiellement sur la gestion des listes chaînées pour simplifier les opérations d'insertion, de supression et de modification des données.

- **Les données des adhérents**

  Comme mentionné ci-dessus, nous avons choisi d'utiliser une liste chaînée afin de simplifier les opérations de données.

  Nous avons donc besoin d'une structure qui stocke les informations d'ahdérent, et par la suite une liste chaînée associée à ces structures.

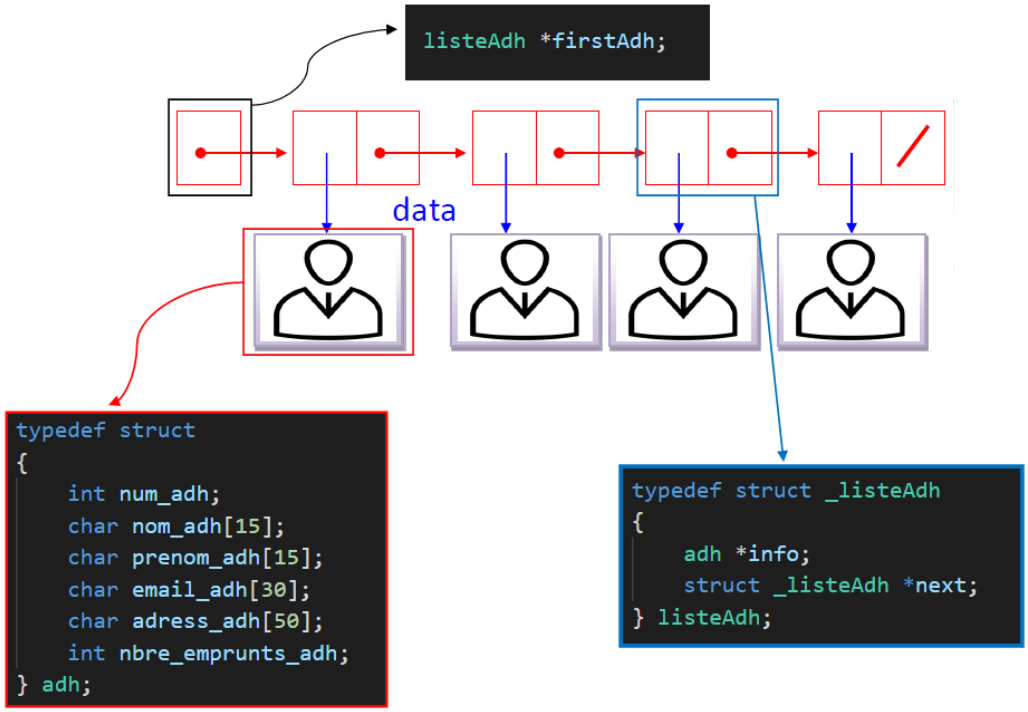

FIGURE 1.2 – Représentation des données des adhérents

- **Les données des livres**

  Comme les données des adhérents, on peut stocker les données des livres dans une liste chaînée, mais on peut optimiser cette manière de représentation : au lieu d'utiliser une seule liste chaînée ; on va utilisé un tableau contenant des structures ayant deux champs :

  — Le premier champ est une chaîne de caractères qui représente la catégorie « *categ_liv* ».
  — Le deuxième est une liste chaînée qui stocke les données des livres de la catégorie « *categ_liv* ».

  Cette manière de représentation est meilleure que la première, car la complexité des opérations sur les données sera réduite.

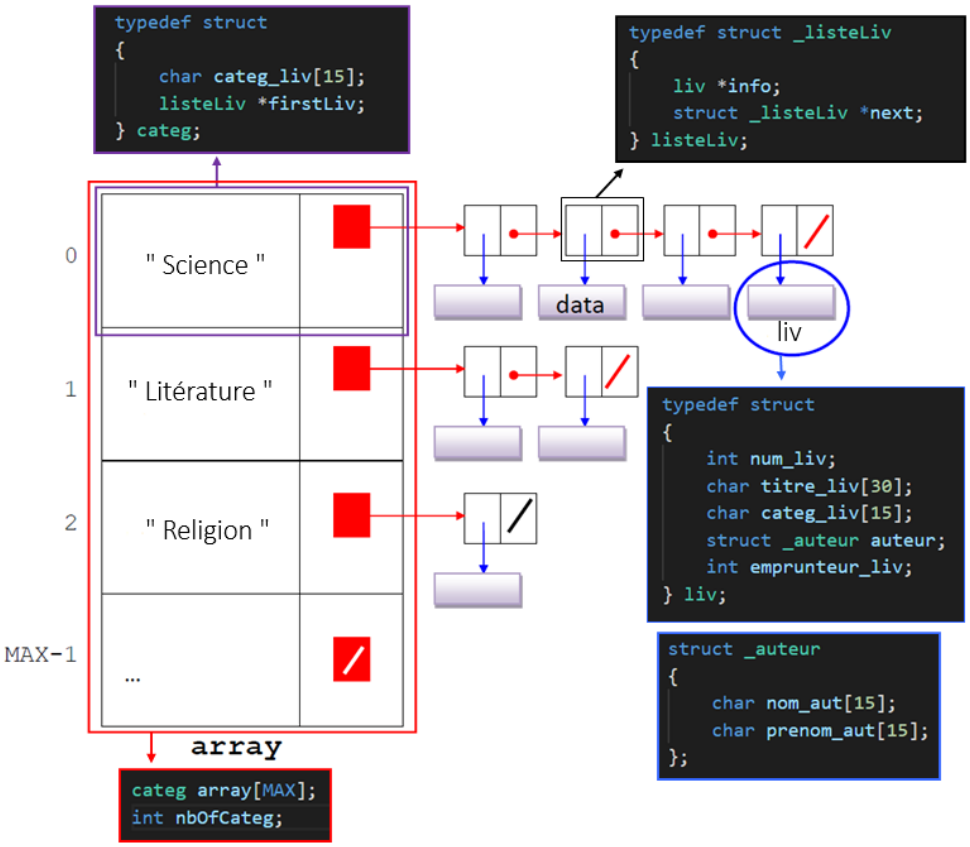

FIGURE 1.3 – Représentation des données des livres

Conclusion

Après avoir dégagé les besoins fonctionnels et opérationnels et tous les critères qu'on doit prendre en considération, et afin de modéliser les besoin attendus de notre application et que les objectifs soient atteinte, on va suivre la démarche du processus unifié qu'on va le détailler dans la prochaine partie.

Chapitre 2

Mise en œuvre et réalisation

Introduction

Dans ce chapitre, nous présentons la partie réalisation et mise en œuvre de notre travail. Pour cela, nous présentons, en premier lieu, l'environnement de travail et les outils de développement utilisés. En second lieu, nous élaborons une présentation des différentes fonctions utilisées.

2.1 Les outils et techniques de développement

2.1.1 Construction de l'application

Visual Studio Code est un éditeur de code source développé par Microsoft pour Windows, Linux et MacOs, il inclut la prise en charge du débogage, du contrôle Git intégré et de GitHub, gratuit, supportant une dizaine de langages dont langage C qu'on a utilisé dans ce projet.

FIGURE 2.1 – Visual Studio Code

2.1.2 Rédaction du rapport

La documentation professionelle nécessite la manipulation du logiciel de traitement de texte **LaTex**. Travailler avec ce dernier est inévitable tôt ou tard, donc nous avons voulu exploiter cette opportunité et explorer **LaTex**.

FIGURE 2.2 – Logiciel LaTex

2.2 Les fonctions de l'application

Pour réaliser ce travail nous avons besoin d'utiliser plusieurs fonctions. Dans cette partie nous allons élaborer chacune des fonctions utilisées dans le code source.

FIGURE 2.3 – Menu principal

Les fonctions de menu principal

- **void gestionAdhs() ;**

  Cette fonction permet de naviguer entre les différentes fonctions qui permettent la gestion des adhérents de la bibliothèque.

FIGURE 2.4 – Gestion des adhérents

- **void gestionLivs() ;**

  Cette fonction permet de naviguer entre les différentes fonctions qui permettent la gestion des livres de la bibliothèque.

FIGURE 2.5 – Gestion des livres

- **void gestionEmpruntes() ;**

  Cette fonction permet de naviguer entre les différentes fonctions qui permettent la gestion des empruntes de la bibliothèque.

FIGURE 2.6 – Gestion des empruntes

- **void quitterProgramme() ;**

  Le rôle de cette fonction est de communiquer avec l'utilisateur pour savoir s'il souhaite enregistrer les modifications.

FIGURE 2.7 – Quitter l'application

Les fonctions des adhérents

- **void afficherAdhs() ;**

  Cette fonction permet de visualiser dans un tableau les informations de tous les adhérents de la bibliothèque.

FIGURE 2.8 – Afficher les adhérents

- **int numUnqiueAdh() ;**

  Cette fonction permet de génerer un numéro unique à l'adhérent d'une manière automatique.

- **void insererAdh() ;**

  Cette fonction demande à l'utilisateur de saisir les informations d'adhérent à ajouter à la liste des ahdérents.

- **void ajouterAdh(adh *info) ;**

  C'est la fonction qui permet d'ajouter l'adhérent à la liste des ahdérents à partir de la structure « *info*» qui contient ses informations.

- **adh *rechercherAdh(char *nom_adh) ;**

  Cette fonction recherche l'adhérent par son nom « *nom_ adh* » et renvoie une structure « *adh* » contenant ses informations.

- **void modifierAdh(char *nom_adh) ;**

  Cette fonction permet de modifier les informations de l'adhérents qui porte le nom « *nom_ adh* ».

- **void supprimerAdh(char *nom_adh) ;**

  Cette fonction permet de supprimer l'adhérent qui porte le nom « *nom_ adh* » de la liste.

Les fonctions des livres

- **void ordonnerLivs() ;**

  Cette fonction permet de visualiser dans un tableau les informations des livres ordonnées par catégorie.

FIGURE 2.9 – Ordonner les livres

- **int numUnqiueLiv() ;**

  Cette fonction permet de génerer un numéro unique au livre d'une manière automatique.

- **void insererLiv() ;**

  Cette fonction demande à l'utilisateur de saisir les informations du livre à ajouter au tableau des listes des livres.

- **void ajouterLiv(adh *info) ;**

  C'est la fonction qui permet d'ajouter le livre au tableau des listes à partir de la structure « *info* » qui contient ses informations.

- **liv *rechercherLiv(char *categ_liv, char *titre_liv) ;**

  Cette fonction recherche le livre par sa catégorie « *categ_liv* » et son titre « *titre_liv* » et renvoie une structure « *liv* » contenant ses informations.

- **void modifierLiv(char *categ_liv, char *titre_liv) ;**

  Cette fonction permet de modifier les informations du livre de catégorie « *categ_liv* » et qui porte le titre « *titre_liv* ».

- **void supprimerLiv(char *categ_liv, char *titre_liv) ;**

  Cette fonction permet de supprimer le livre de catégorie « *categ_liv* » et qui porte le titre « *titre_liv* ».

Les fonctions des empruntes

- **void emprunterLiv(char *nom_adh, char *categ_liv, char *titre_liv) ;**

  C'est une fonction pour emprunter un livre de catégorie « *categ_liv* » et de titre « *titre_liv* » par un adhérent du nom « *nom_adh* ».

- **void afficherLivsEmpruntes() ;**

  Cette fonction permet de visualiser dans un tableau les livres empruntés.

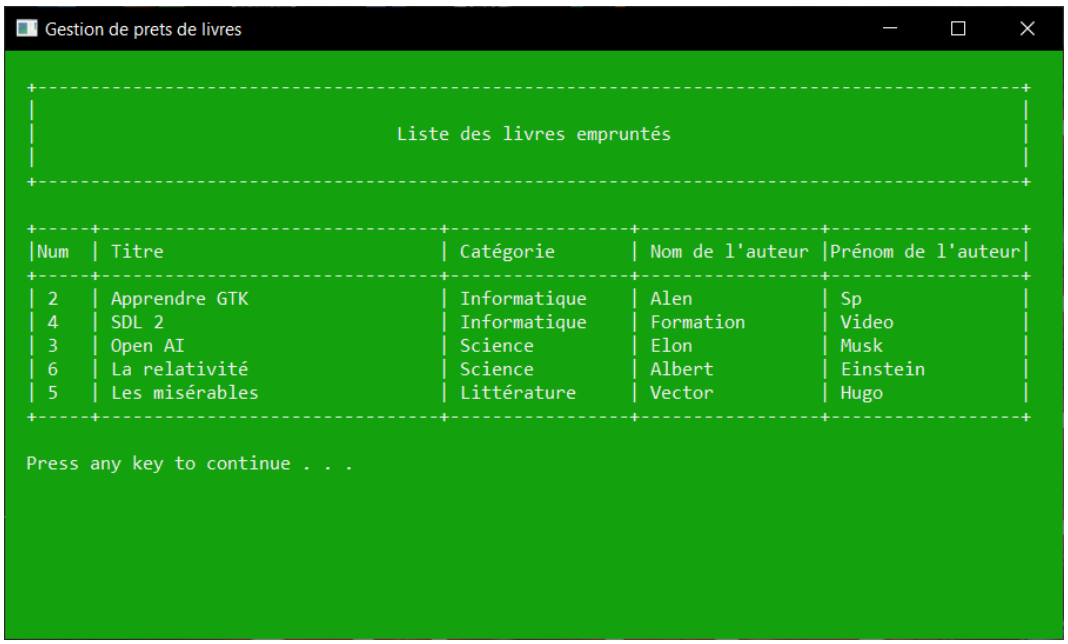

FIGURE 2.10 – Afficher les livres empruntés

- **void afficherAdhEmprunteurs(char *categ_liv, char *titre_liv) ;**

  Cette fonction permet d'afficher les emprunteurs de livre de catégorie « *categ_liv* » et de titre « *titre_liv*».

- **void rendreLiv(char *nom_adh, char *categ_liv, char *titre_liv) ;**

  C'est une fonction pour rendre le livre de catégorie « *categ_liv* » et de titre « *titre_liv* » qui est emprunté par l'adhérent du nom « *nom_adh* ».

Les fichiers

Pour enregistrer les informations des adhérents et des livres, on a besoin des fichiers binaires (ou textes) pour stocker ces informations. Pour cela, nous avons utilisé deux fichiers binaires ***adherents.dta*** et ***livres.dta***.

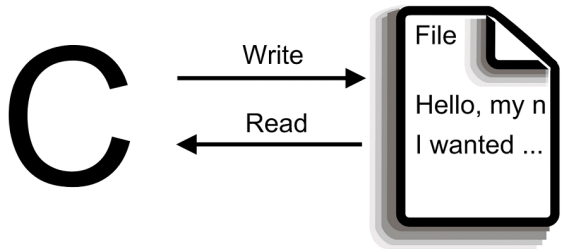

FIGURE 2.11 – Charger et sauvegarder

Pour atteindre cet objectif, nous avons défini des fonctions pour effectuer ces tâches :

- **void init() ;**

  Cette fonction permet d'initialiser la liste des adhérents et le tableau des listes des livres pour les charger à partir des fichiers.

- **void chargerInfos() ;**

  Le rôle de cette fonction est le charge des informations des adhérents et des livres à partir des deux fichiers *adherents.dta* et *livres.dta*.

  ⇒Cette fonction est appelée une seule fois au début du programme.

- **void sauvegarderInfos() ;**

  Le rôle de cette fonction est le sauvegarde des informations des adhérents et des livres sur les deux fichiers *adherents.dta* et *livres.dta*.

  ⇒Cette fonction est appelée une seule fois à la fin du programme (dans la fonction *quitterProgramme()*).

2.3 Organisation du projet

Nous avons réparti le programme sur plusieurs fichiers afin de le rendre **plus clair** et **plus lisible**.

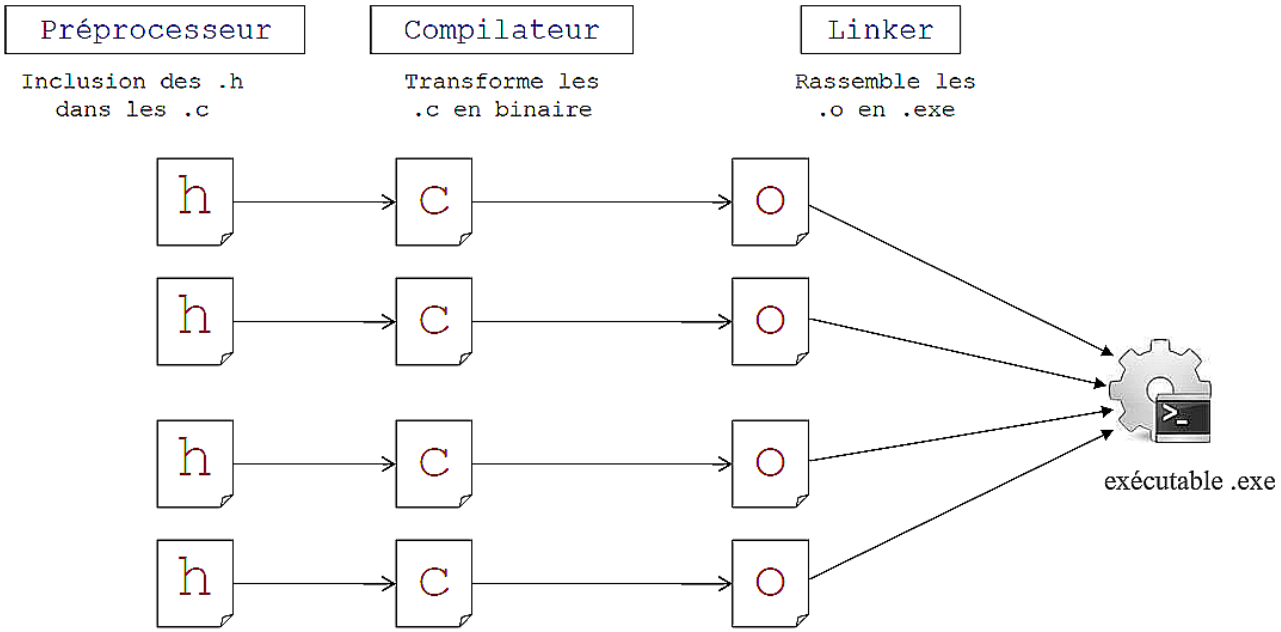

FIGURE 2.12 – Programmation modulaire

Pour y parvenir, nous avons réparti notre programme en cinq fichiers (**.c**) et un fichier en-tête (**.h**) (*head-file*) (**programmation modulaire**) :

- **main.c**

  Ce fichier contient le menu principal du programme et tous les fonctions de ce menu (*gestionAdhs(), gestionLivs(), gestionEmpruntes(), quitterProgramme()*).

- **adherents.c**

  Ce fichier contient les fonctions qui permettent la gestion des adhérents (*ajouterAdh(adh *info), adh *rechercherAdh(char *nom_adh), ...*).

- **livres.c**

  Ce fichier contient les fonctions qui permettent la gestion des livres (*ajouterLiv(liv *info), liv *rechercherLiv(char *categ_liv, char *titre_liv), ...*).

- **empruntes.c**

  Ce fichier contient les fonctions qui permettent la gestion des empruntes (*emprunterLiv(char *nom_adh, char *categ_liv, char titre_liv), ...*).

- **fichiers.c**

  Ce fichier contient les trois fonctions qui interagissent avec les fichiers **adherents.dta** et **livres.dta** (*void init(), void chargerInfos(), void sauvegarderInfos()*).

- **declarations.h**

  Ce fichier contient les éléments que devra connaître un utilisateur du module (*définitions des constantes, Macro-instructions, déclarations des structures, prototypes des fonctions définies dans les fichiers (**.c**)*). Ce fichier constitue l'interface du module et doit être inclut dans tous les fichiers (**.c**) .

Conclusion générale

Notre projet a consisté à la réalisation d'une application en **langage C** pour **la gestion de prêts de livres** à la bibliothèque **ENSIAS**. Ce projet nous a permis d'approfondir nos connaissances théoriques acquises le long de notre formation, par la pratique des nouvelles technologies. Cette expérience nous a permis de maîtriser **le langage C** et particulièrement « *les structures de données* » (*listes chaînées*), « *la gestion des fichiers* » et « *la programmation modulaire* ».

Ressources

[1] Le cours de « structure de données » de **Monsieur EL FAKER**.

[2] Le cours de « Technique de programmation » de **Monsieur Hatim GUERMAH**.

[3] <https://www.youtube.com/watch?v=_Z39xKbRd2E>. [Tutoriel C - fichiers binaires].

[4] <https://stackoverflow.com>.

[5] <https://www.overleaf.com/learn/>.

[6] <https://fr.wikipedia.org//>.