Xavier MAYER

Projet RUST

Présentation du projet :

Il s'agit de la bibliothèque demandée dans l'énoncé. Comme dans l'énoncé, on peut ajouter un livre, emprunter un livre via son titre, retourner un livre toujours par son titre, afficher tous les livres, afficher seulement les livres disponibles et quitter le programme.

Choix techniques :

Tout d'abord, j'ai hésité à ajouter une option modifier le livre en fonction de son titre comme pour l'ajout, l'emprunt et le retour. On utiliserait la même méthode qui nous permet de trouver le livre lorsqu'on veut l'emprunter (iter _mut().find(«partie tapée ») et io ::stdin().read_line(« partie tapée »)), la conversion et le match puis le mutable.

Ensuite, j'ai fait le choix de ne bloquer que les entrées différentes de chiffre pour la date, je pensais tout d'abord à bloquer et forcer un affichage YYYY mais je me suis dit que des écrits existent avant l'an 1000 et que je ne pouvais pas bloquer leurs ajouts.

Voici pour mes choix, je vais décrire ce que j'ai pu utiliser dans mon code :

- Struct me permet de définir ce que contient mon livre, un titre (en string), un auteur (en string), l'année (en int) et un booléen pour la disponibilité (disponible ou emprunté). (Comme demandé dans l'énoncé, j'ai juste pris la liberté pour les champs, titre, auteur et année. Comme ils n'étaient pas imposé j'ai choisi les types les plus cohérents à mon sens.)

- Je vais gérer les potentielles erreurs de conversion grâce aux « parse » dans mon code.

- Je me protège des doublons avec iter().any(------). iter me sert à boucler et j'utilise any + le paramètre comme closure pour éviter d'avoir à faire une nouvelle boucle.

- Il y a bien une fonction par option : ajouter_livre, emprunter_livre, retourner_livre, afficher_tous_les_livres et afficher_livres_disponibles. Le quitter est dans le main.

- J'utilise use std::io; pour gérer les entrées claviers de l'utilisateur (lecture ect.).

- par exemple io::stdin() me permet de taper, read_line(&mut titre) me permet de lire ce qui a été écrit et le stocker, &mut titre afin que Rust puisse modifier ce qui a été tapé et unwrap() pour la partie erreur. Ensuite j'utilise trim pour gérer les espaces et retours à la ligne, ça m'évite les erreurs pour les comparaisons.

- J'utilise push comme vu en cours pour que les données entrées soit ajoutées dans mon vecteur.

- J'utilise match pour le choix, pour faire des checks, je l'utilise avec find, pour qu'il me renvoie soit le livre cherché, soit le fait que le livre n'existe pas. Pareil je l'utilise pour vérifier si la conversion des dates est réussie pour que le programme soit plus résistant à la casse.

- J'utilise une boucle for pour afficher les livres, à la fois dans la fonction tous les livres mais aussi seulement les livres disponibles. Il y a juste une condition en plus dans le

deuxième.cas.

- Le programme menu boucle bien grâce à la loop dans le main, le break est trigger par le choix 6.Quitter ou ctrl + c