Mini-Projet de programmation

Il s'agit de produire un programme (fichiers source et un exécutable testé et opérationnel avec la documentation), dont le sujet est présenté dans la suite du document.

Ce qu'il faut rendre :

- Le programme source complet largement commenté
- Un exécutable testé et opérationnel avec sa documentation d'installation
- Un rapport présentant le travail effectué :
  1. Organisation du programme : découpage en fonctions, rôle de ces fonctions, explications du programme
  2. Mode d'emploi du programme
  3. Bilan qualitatif du travail, difficultés rencontrées, etc.

Le tout sera placé dans un fichier archive[1], à envoyer par mail à votre enseignant de groupe.

Évaluation du projet

Elle repose sur les éléments suivants :

- Le programme source :
  - Respect de l'énoncé, originalité du thème proposé
  - Qualité technique du programme : découpage en fonctions, instructions, algorithmes, efficacité, gestion des erreurs…
- Présentation du programme : indentation, commentaires et nommage
- Documentation fournie
  - Organisation du programme et son mode d'emploi
  - Bilan
- Soutenance du travail
  - Démonstration du programme
  - Interrogation individuelle sur le travail réalisé

Présentation du projet

Vous devrez réaliser un programme permettant de gérer un système de réservation d'*objets* quelconques : livres, cassettes vidéo, chambres d'hôtel, voiture, ressources d'une école (salles de cours, enseignements...), etc. Les exemples donnés ci-dessous concernent une gestion de

prêts de livres dans une bibliothèque : vous devrez personnaliser votre programme en fonction de votre choix.

A vous de choisir un système de gestion : soyez imaginatifs et créatifs

Le modèle à mettre en œuvre contiendra deux structures différentes, dont les champs contiennent à minima les informations données ci-dessous. Vous devrez sans doute ajouter des champs dans ces structures en fonction du système de réservation que vous avez choisi.

ADHERENT	LIVRE
typedef struct SAdherent { int adh_Index; char adh_Nom[CMAX]; int adh_NbEmprunts; } Adherent;	typedef struct SLivre { int liv_Index; char liv_Titre[CMAX]; char liv_Auteur[CMAX]; int liv_Emprunteur; } Livre;

adh_Index (resp. **liv_Index**) identifie de manière unique un adhérent (resp. un livre). Il vous appartient de gérer la notion de numéro unique. **liv_Emprunteur** contient la valeur de **adh_Index** de l'emprunteur du livre. Le nombre de livres empruntés par un adhérent est stocké dans **adh_NbEmprunts** (un adhérent peut donc emprunter plusieurs livres).

Contraintes de programmation

La fonction **main()** devra contenir un tableau de structures **Adherent** et un tableau de structure **Livre** (ce ne seront pas des variables globales). Vous devrez enregistrer les données de ces structures dans deux fichiers, en utilisant **fscanf()** et **fprintf()**.

Votre programme devra proposer un menu comme celui-ci :

(1) Gestion des adhérents
- Ajouter, modifier ou supprimer un adhérent
- afficher la liste des adhérents par ordre alphabétique

(2) Gestion des livres
- Ajouter, modifier ou supprimer un livre
- afficher la liste des livres par ordre alphabétique (titre)

(3) Gestion des emprunts
- Emprunter un livre
- Rendre un livre
- Afficher la liste des livres empruntés
- Afficher la liste des emprunteurs de livres

(4) Quitter le programme

Notes de mise en œuvre

La *suppression* d'un élément du tableau implique de décaler la « *case libre* » vers la fin du tableau afin de supprimer les « *trous* » du tableau.

Tous les tableaux doivent être passés par paramètre de la manière suivante :
- Si la fonction ne modifie pas le nombre **borne** d'éléments présents dans le tableau **T** dont le nombre maximal est **taille**, elle devra être déclarée comme suit :
  <type> fonctionUtiliseTableau (<type> T[ ], int borne, int taille) ;
- Si la fonction modifie le nombre, pointé par **pborne**, d'éléments présents dans le tableau **T** dont le nombre maximal est **taille**, elle devra être déclarée comme suit :
  <type> fonctionModifieTableau (<type> T[ ], int *pborne, int taille)
  Au retour de la fonction, ***pborne** donne le nouveau nombre d'éléments présents.