Examen de Langage et logique de programmation 2, labo

Enoncé

L'examen consiste à écrire un programme C (une application) fonctionnel en binôme (avec un coéquipier) et répondre individuellement aux questions posées par le professeur sur ce programme. Il faudra éventuellement montrer le programme en fonctionnement. Le programme sera découpé en ***deux phases*** :

- Phase 1 : Construction d'une liste linéaire dynamique (avec sous-listes) contenant les informations les plus pertinentes[1] obtenues en lisant les fichiers ;

- Phase 2 : Affichage d'un menu comportant différentes actions et exécution de la tâche choisie par l'utilisateur en utilisant prioritairement la liste linéaire dynamique plutôt que les fichiers.

Description des fichiers

L'application manipulera trois fichiers :

- Deux fichiers binaires (A et B) qui permettent de retenir 2 listes distinctes de données. A vous de décider quelles données, mais le fichier A ne doit pas contenir plus de 4 champs de données (4 « colonnes ») et par contre le fichier B devra obligatoirement contenir plus de 4 champs. Les données sont structurées et donc, on doit définir des « struct » pour chaque fichier. Il faudra au moins un champ « id » qui contiendra un entier unique permettant d'identifier chaque donnée ;

- Un fichier binaire (X) qui contient la liste des correspondances entre les données du fichier A et les données du fichier B. Il pourra éventuellement contenir des *données (« colonnes ») supplémentaires* (maximum 2 !) pour chaque association d'un élément de A à un élément de B. On doit également définir une « struct » pour ce fichier.

Description de la phase 1 du programme (en tout début de programme, exécuté à chaque lancement de celui-ci)

Pour chaque fichier, si celui-ci n'existe pas, un fichier vide sera créé.

Ensuite, ces 3 fichiers seront consultés de manière à construire une liste principale d'éléments non supprimés du fichier A, et pour chaque élément de cette liste principale, on trouve un pointeur vers une liste secondaire (ou « sous-liste ») d'éléments non supprimés du fichier B qui sont liés à cet élément A.

La liste principale sera dynamique (via malloc() et avec un pointeur de tête et des pointeurs « next ») et constituée des éléments non supprimés du fichier A. Elle sera triée selon le tri que vous estimez le plus judicieux pour votre application. Il faudra définir une « struct » propre à cette liste principale permettant de chaîner les enregistrements de A et de pointer également vers les sous-listes.

Chaque élément de la liste A principale contiendra également un pointeur vers une liste secondaire (une « sous-liste ») dynamique (via malloc() et avec un pointeur de tête et des pointeurs « next ») contenant les éléments de la liste B (non supprimés) qui sont en relations avec cet élément de liste principale. Les relations entre A et B se trouvent bien entendu dans le troisième fichier (X). Si certaines *données supplémentaires* (2 max) ont été ajoutées à X, elles peuvent éventuellement se retrouver dans les éléments des listes secondaires B. **Les éléments des liste secondaires ne devront en aucun cas contenir toutes les informations du fichier B**. (Certaines infos – des « colonnes » – qui se retrouvent dans le fichier B ne se retrouveront donc pas dans ces sous-listes.) Il faut donc définir une « struct » propre à ces sous-listes qui contiendra les champs que l'on gardera en mémoire dynamique

1 Les infos essentielles (dont les identifiants des enregistrements), d'usage fréquent, qui seront accessibles directement.

ainsi qu'un champ offset, de type long, qui permettra via un fseek() de lire le bon enregistrement directement pour obtenir toutes les données de B si nécessaires.

Description de la phase 2 du programme (un menu en boucle juste après la phase 1)

Le logiciel proposera ensuite un menu unique qui permettra de choisir parmi les actions suivantes :

- Ajout de A : Permet d'ajouter un nouvel élément de A encodé via des saisies contrôlées avec getchar() à la fois dans le fichier et dans la liste dynamique (en tenant compte du tri). Dans le fichier A, un enregistrement doit être ajouté en fin de fichier tandis que dans la liste principale A, il faut insérer le nouvel élément au bon endroit dans la liste car elle est triée ;

- Ajout de B : Permet d'ajouter un nouvel élément de B encodé via des saisies contrôlées avec getchar() à la fois dans les fichiers et dans les sous-listes dynamiques. Dans les fichiers B et X, les enregistrements doivent être ajoutés en fins de fichiers, *en mémorisant l'offset dans B pour plus tard*, et dans **chaque** sous-listes des éléments de A associés à cet élément de B, il faut insérer le nouvel élément de B en <u>début</u> de sous-liste (<u>LIFO</u>) *avec l'offset mémorisé précédemment*.
  ATTENTION : Ce menu doit permettre :
  1) D'encoder le nouvel élément de B ;
  2) Puis d'ajouter B en fin de fichier ;
  3) Puis, **en boucle** :
     a. Permettre à l'utilisateur d'encoder un (ou aucun) identifiant de A (non supprimé) juste après l'affichage d'une liste de propositions (affichée de manière claire et conviviale). Cette saisie doit être contrôlée ;
     b. Et **si un A correspondant a été choisi :**
        i. Insérer l'élément B en début de sous-liste de l'élément de A choisi ;
        ii. Ajouter une ligne (un enregistrement) en fin de fichier X (pour lier le nouvel élément B à l'élément de A choisi).
     c. ATTENTION : A chaque tour de boucle, les éléments de A choisis précédemment doivent disparaître des propositions (pour éviter l'encodage de doublons) ;

- Suppression de A : Permet de supprimer un élément aussi bien dans les fichiers que dans la liste dynamique. Dans les fichiers (A et X) on fera des suppressions logiques. Dans la liste principale A, on sort l'élément du chaînage et on le libère ;

- Suppression de B : Permet de supprimer un élément aussi bien dans le fichier que dans la liste dynamique. Dans les fichiers (B et X) on fera des suppressions logiques. Dans les sous-listes, on supprime les éléments B correspondants de toutes les sous listes ! ;

- Listage : Affiche (de manière conviviale et claire) toutes les infos de la liste principale et des listes secondaires. Ces infos sont donc triées et **aucun accès au fichier n'est permis ici** ;

- Affichage détaillé : L'utilisateur saisit un identifiant de A et le logiciel affiche toutes les données correspondantes (de A et des B liés) en faisant le moins possible de lectures dans les fichiers. Pour y parvenir, on utilisera le fait que la « struct » de B pour les sous-listes dynamiques contient un champ « offset » qui donne le numéro d'octet (de type long) depuis le début du fichier permettant d'arriver à l'enregistrement correspondant dans le fichier B via fseek() ;

- Toute autre fonctionnalité est facultative mais bienvenue ! (BONUS) ;

- Quitter : Quitte « proprement » le logiciel.

Evaluations (découpe du travail)

1. Une première évaluation aura lieu le **vendredi 22 mars 2024**. Vous devrez montrer un programme en développement **sans fichiers ni listes dynamiques** qui permet d'encoder toutes les données de manière sécurisées[2] dans deux vecteurs de structures (un pour les éléments de A, un pour les éléments de B). Voici en détail, ce que le logiciel permet de faire et comment :
    a. Phase 1 : / ;
    b. Phase 2 : Le programme propose un menu unique qui permet de choisir parmi les actions suivantes pour chaque liste :

        i. Ajout de A : permet d'ajouter un élément de A encodé via des saisies contrôlées avec getchar() en fin de vecteur A correspondant. S'il n'y a plus de cellule disponible (vecteur rempli), le programme affiche un message d'erreur et n'effectue pas d'ajout ;

        ii. Ajout de B: permet d'ajouter un élément de B encodé via des saisies contrôlées avec getchar() en fin de vecteur B correspondant. S'il n'y a plus de cellule disponible (vecteur rempli), le programme affiche un message d'erreur et n'effectue pas d'ajout ;

        iii. Suppression de A : permet de supprimer un élément dans le vecteur A. (On fera une suppression logique.) ;

        iv. Suppression de B : permet de supprimer un élément dans le vecteur B. (On fera une suppression logique.) ;

        v. Listage des A: On affiche tous les éléments de A (dans le vecteur A) non supprimés contenus dans le vecteur A ;

        vi. Listage des B: On affiche 3 champs max (dont l'identifiant) de tous les éléments de B (dans le vecteur B) non supprimés contenus dans le vecteur B ;

        vii. Affichage détaillé d'un B : L'utilisateur saisit un identifiant de B et le logiciel recherche et affiche toutes les données correspondantes de B ou un message indiquant l'absence de B ;

        viii. Quitter : Quitte le logiciel.

2. Une deuxième évaluation aura lieu le **lundi 15 avril 2024**. Vous devrez montrer un programme en développement **sans listes dynamiques** qui permet d'encoder toutes les données de manière sécurisées[2] dans 3 fichiers (A, B et X). Voici en détail, ce que le logiciel permet de faire et comment :
    a. Phase 1 : Pour chaque fichier, si celui-ci n'existe pas, un fichier vide sera créé ;
    b. Phase 2 : Le programme propose un menu unique qui permet de choisir parmi les actions suivantes pour chaque liste :

        i. Ajout de A : Permet d'ajouter un nouvel élément de A encodé via des saisies contrôlées avec getchar() en fin de fichier A ;

        ii. Ajout de B : Permet d'ajouter un nouvel élément de B encodé via des saisies contrôlées avec getchar() dans les fichiers B et X, les enregistrements doivent être ajoutés en fins de fichiers.
        ATTENTION : Ce menu doit permettre :
            1. D'encoder le nouvel élément de B ;
            2. Puis d'ajouter B en fin de fichier ;
            3. Puis, en boucle :

[2] avec des fonctions de saisies contrôlées via des getchar()

a. Permettre à l'utilisateur d'encoder un (ou aucun) identifiant de A (non supprimé) juste après l'affichage d'une liste de propositions (affichée de manière claire et conviviale). Cette saisie doit être contrôlée ;
b. Et si un A correspondant a été choisi :
    - Ajouter une ligne (un enregistrement) en fin de fichier X (pour lier le nouvel élément B à l'élément de A choisi).
c. ATTENTION : A chaque tour de boucle, les éléments de A choisis précédemment doivent disparaître des propositions (pour éviter l'encodage de doublons) ;

iii. Suppression de A : permet de supprimer un élément dans le fichier A. (On fera une suppression logique.) ;

iv. Suppression de B : permet de supprimer un élément dans le fichier B. (On fera une suppression logique.) ;

v. Listage : Affiche (de manière conviviale et claire) toutes les données des éléments de A, ensuite pour chaque élément de A, on affiche 3 champs max (dont l'identifiant) des éléments de B qui sont liés à A (en consultant le fichier X)[3] ;

vi. Affichage détaillé d'un B : L'utilisateur saisit un identifiant de B et le logiciel recherche (dans le fichier B) et affiche toutes les données correspondantes de B ou un message indiquant l'absence de B ;

vii. Quitter : Quitte le logiciel.

3. Une troisième et dernière évaluation aura lieu le **mardi 14 mai 2024**. Vous devrez montrer un programme complet qui répond à l'énoncé initial (en pages 1 et 2).

Bon travail tous !

[3] Les lignes de code qui mettent en œuvre ce listage permettront de servir de bon point de départ pour la phase 1 de la version finale du programme

Annexe :

Vous trouverez ci-dessous un exemple de contenu des fichiers.

Fichier A : **abonnes.dat**

ID	Pseudo	NavigParDef	Date_Abo
1	Dubuis0485	Chrome	10/01/2020
2	IThinkIloveC	Chrome	23/12/2021
-3	Machin0807	Edge	15/11/2023
4	Leg@Pol	Opera	03/03/2023
5	AaaAAaa-AA	Safari	06/10/2022
6	Minouchette08	Safari	12/12/2020
7	SingularitéPrimordiale	Edge	20/11/2023
...			

Fichier B : **videos.dat**

(Offset)	ID	Url	NBVues	Titre	Date-Heure-Pub
(0)	1	https://www.youtube.com/watch?v=btDySlFQCew	5648	Salut tout le monde !	10/11/2020 à 12h35
(250)	2	https://www.youtube.com/watch?v=nnFijLHX7TM	15412	Je scie une brique avec mes doigts !	10/12/2020 à 8h30
(500)	3	https://www.youtube.com/watch?v=AzdRSSWY55o	915465	Je casse une noix avec mes fesses !	05/01/2020 à 16h40
(750)	4	https://www.youtube.com/watch?v=Nta93O-mij8	124129	Je fais léviter ma grand-mère !	17/01/2020 à 16h30
(1000)	5	https://www.youtube.com/watch?v=R6NSqmiv7fQ	52654	J'épluche 8 concombres en 1 minute !	19/01/2020 à 16h30
(1250)	6	https://www.youtube.com/watch?v=OEXCZV0qDz4	87456	Je déballe une chupa-chups avec ma bouche !	20/01/2020 à 10h30
(1500)	7	https://www.youtube.com/watch?v=9UAClc-CXYU	112456	Je déballe un ferrero avec mes pieds !	22/01/2020 à 8h45
...	...				

Fichier C : **likes.dat**

ID_abo	ID_video	Date_like
1	3	05/01/2020
1	5	19/01/2020
1	1	13/11/2020
2	5	29/02/2020
2	7	29/02/2020
-3	1	15/11/2021
-3	2	15/11/2021
4	4	03/04/2022
6	2	15/12/2022
6	5	20/01/2023

7	1	20/11/2023
7	2	20/11/2023
7	3	20/11/2023
7	4	20/11/2023
7	6	20/11/2023

NB :

- Par facilité, les différents enregistrements des fichiers, bien qu'ils soient conservés les uns après les autres sur le disque, seront présentés les uns en-dessous des autres dans cet exemple ;
- Les dates et heures sont présentées librement par souci de lisibilité ;
- A gauche du fichier B, on montre des exemples d'offset correspondant à chaque enregistrement, il ne s'agit pas de données qui sont stockées dans les fichiers ;
- Les id négatifs sont des suppressions logiques.

Vous trouverez ci-dessous un exemple de début de liste dynamique (principale et secondaires) correspondant à l'exemple donné plus haut :