NF16 - TP 3 – Listes chaînées

Introduction

Nous proposons d'écrire un programme permettant la gestion d'un magasin et des rayons qui le constituent : un magasin est une entité contenant plusieurs rayons, et chaque rayon contient plusieurs produits.

L'ensemble des données et des liens entre ces données seront représentées sous forme de listes simplement chaînées.

A. Structures

1. Créer une structure **`Produit`** (et le type correspondant **`T_Produit`**) ayant les champs suivants :
   - **`marque`** *de type chaine de caractères*
   - **`prix`** *de type nombre réel*
   - **`qualite`** *de type caractère (valeurs possibles : A, B ou C)*
   - **`quantite_en_stock`** *de type entier*
   - **`suivant`** *de type pointeur vers une structure Produit*

2. Créer une structure **Rayon** (et le type correspondant **`T_Rayon`**) ayant les champs suivants :
   - **`nom_rayon`** *de type chaine de caractères*
   - **`nombre_produits`** *de type entier (nombre de marques de produit distinctes dans le même rayon; on considère que l'on ne peut pas avoir deux produits de la même marque dans un même rayon)*
   - **`premier`** *de type pointeur vers une structure Produit*
   - **`suivant`** *de type pointeur vers une structure Rayon*

3. Créer une structure **`Magasin`** (et le type correspondant **`T_Magasin`**) ayant les champs suivants :
   - **`nom`** *de type chaine de caractères*
   - **`premier`** *de type pointeur vers une structure Rayon*

B. Fonctions requises

1. Création et initialisation des structures :

   ```
   T_Produit *creerProduit(char *marque, float prix, char qualite, int quantite)
   T_Rayon *creerRayon(char *nom)
   T_Magasin *creerMagasin(char *nom)
   ```

2. Ajout d'un rayon dans un magasin :

   ```
   int ajouterRayon(T_Magasin *magasin, T_Rayon *rayon)
   ```

   renvoie 1 si l'ajout s'est bien passé, 0 sinon; l'ajout se fait en respectant le tri par ordre alphabétique sur le nom du rayon; on ne doit pas autoriser l'utilisateur à ajouter deux fois le même rayon

3. Ajout d'un produit dans un rayon :

   ```
   int ajouterProduit(T_Rayon *rayon, T_Produit *produit)
   ```

renvoie 1 si l'ajout s'est bien passé, 0 sinon; l'ajout se fait en respectant le tri par ordre croissant du prix du produit; on ne doit pas autoriser l'utilisateur à ajouter deux fois la même marque de produit dans un rayon

4. Affichage de tous les rayons d'un magasin :

   `void afficherMagasin(T_Magasin *magasin)`

   L'affichage se fait sous forme de liste triée sur le nom des rayons

Nom	Nombre de produits
Boissons	15
Lessives	7
Yaourts	11

5. Affichage de tous les produits d'un rayon :

   `void afficherRayon(T_Rayon *rayon)`

   L'affichage se fait sous forme de liste triée sur le prix du produit

Marque	Prix	Qualité	Quantité en stock
Lavazza	3.50	A	100
Carte Noire	4.20	C	80
Grand-Mère	4.80	B	55

6. Suppression d'un produit dans un rayon :

   `int supprimerProduit(T_Rayon *rayon, char* marque_produit)`

   renvoie 1 si le retrait s'est bien passé, 0 sinon; on veillera à libérer la mémoire précédemment allouée

7. Suppression d'un rayon et de tous les produits qu'il contient :

   `void supprimerRayon(T_Magasin *magasin, char *nom_rayon)`

   renvoie 1 si la suppression s'est bien passée, 0 sinon; on veillera à libérer la mémoire précédemment allouée

8. Recherche des produits se situant dans une fourchette de prix entrée par l'utilisateur :

   `void rechercheProduits(T_Magasin *magasin, float prix_min, float prix_max)`

   L'affichage se fait sous forme de liste triée par ordre croissant de prix du produit.

Marque	Prix	Qualité	Quantité en stock	Rayon
Carte Noire	2.50	A	100	Café
Nutella	3.10	B	250	Petit Déjeuner
Danone	3.50	B	80	Yaourts

   Vous expliquerez dans votre rapport votre choix de créer des structures spécifiques ou d'utiliser les structures déjà définies pour cette fonction. ***Vous veillerez à optimiser cette fonction de manière à effectuer le minimum d'opérations possible.***

9. Question Bonus : fusionner deux rayons

   `void fusionnerRayons(T_Magasin *magasin)`

L'utilisateur choisit deux rayons parmi ceux du magasin. On fusionne alors ces deux rayons en un seul dont le nom sera au choix de l'utilisateur et qui contiendra les produits des deux rayons fusionnés triés par ordre croissant de prix. ***Vous veillerez à optimiser cette fonction de manière à effectuer le minimum d'opérations possible.***

C. Programme Principal :

Programmer un menu contenant les options suivantes :

1. Créer un magasin
2. Ajouter un rayon au magasin
3. Ajouter un produit dans un rayon
4. Afficher les rayons du magasin
5. Afficher les produits d'un rayon
6. Supprimer un produit
7. Supprimer un rayon
8. Rechercher un produit par prix
9. Quitter

Consignes générales :

- Sources

L'organisation de votre projet sera la suivante:

- Fichier d'entête **tp3.h** contenant les définitions de types, de structures, de constantes et les prototypes de vos fonctions
- Fichier source **tp3.c** contenant les définitions de vos fonctions
- Fichier source **main.c** contenant le programme principal

- Rapport

Votre rapport de quatre pages maximum contiendra :

- La liste des structures et des fonctions supplémentaires que vous aurez choisi d'implémenter et les raisons de ces choix
- Un exposé succinct de la complexité de chacune des fonctions implémentées

Votre rapport et **vos trois fichiers** feront l'objet d'une remise de devoir sur **Moodle** dans l'espace qui sera ouvert à cet effet pendant une semaine suivant votre démonstration au chargé de TP (un seul rendu de devoir par binôme).