TP n° 6

I3 : Listes chaînées

Exercice 1 [Liste chaînée d'articles]

On considère la liste d'articles en vente dans un supermarché. Chaque article est désigné par sa référence (le numéro de code barre) et il a un prix. La référence est un nombre entier, le prix est un nombre décimal à virgule flottante.

1. Définissez un type de données `article_t` qui permette de représenter un article.
2. On souhaite conserver l'ensemble des articles du magasin dans une liste chaînée. Définissez un type de données `liste_article_t` permettant de mettre un article dans une liste chaînée. Chaque élément doit contenir les informations sur l'article (prix et référence), et un pointeur vers l'élément suivant.
3. Le dernier élément d'une liste chaînée doit pointer vers NULL. C'est ce qui marque la fin de la liste. Lorsqu'une liste est vide, elle est uniquement constituée d'un pointeur à la valeur NULL. Écrivez une fonction `creerListe()` qui retourne un pointeur vers une liste vide.
4. Écrivez une fonction `creerArticle()` qui prend en paramètres le prix et la référence d'un article et retourne un pointeur vers un nouvel élément de type `liste_article_t`.
5. Écrivez une procédure `afficherArticle()` qui prend comme paramètre un pointeur vers un article et affiche les informations sur cet article (son prix et sa référence).
6. Écrivez une fonction `enlisterDebut()` qui ajoute un article au début d'une liste chaînée. Cette fonction prend en paramètres un pointeur vers le début de la liste et un pointeur vers le nouvel article à ajouter, et retourne un pointeur vers le nouveau début de la liste.
7. Écrivez une fonction `nbElements()` qui prend en paramètre un pointeur vers le début d'une liste chaînée et retourne le nombre d'éléments contenus dans cette liste.
8. Écrivez une procédure `afficherListe()` qui affiche un par un les articles contenus dans une liste chaînée. Cette procédure prend en paramètre un pointeur vers le début de la liste.
9. Écrivez une fonction `viderListe()` qui prend en paramètre un pointeur vers une liste et la vide en libérant tous ses éléments. La fonction retourne un pointeur vers un élément de la liste à la valeur NULL.
10. Écrivez un programme qui crée une liste vide, puis crée quelques articles et les met au début de la liste. Le programme affiche ensuite le nombre d'éléments dans la liste puis tous les éléments un par un. À la fin du programme la liste est vidée et le programme se termine.
11. Écrivez une fonction `retirerArticle()` qui prend en paramètres un pointeur vers un élément et un pointeur vers le début d'une liste chaînée, et supprime cet élément de la liste. La fonction retourne un pointeur vers la liste.
12. Écrivez une fonction `retirerRef()` qui supprime de la liste le ou les élément(s) correspondant à une référence passée en paramètre d'une liste dont le pointeur vers le premier élément est passé en paramètre, et retourne un pointeur vers la liste. Vous pourrez utiliser la fonction `retirerArticle()` écrite à la question précédente.

Exercice 2 [Liste chaînée triée]

Cet exercice réutilise des structures de données et des fonctions qui ont été écrites dans l'exercice précédent.

1. Écrivez une fonction `enlisterTriPrix()` qui prend en paramètres un pointeur vers un nouvel élément et un pointeur vers une liste chaînée, insère le nouvel élément dans la liste chaînée de façon triée par ordre croissant de prix, et retourne un pointeur vers la liste.
2. Écrivez une fonction `copierArticle()` qui prend en paramètre un pointeur vers un élément d'une liste d'article (type `liste_article_t*` et copie cet article dans un nouvel élément à l'exception de son pointeur. Le pointeur du nouvel élément prend la valeur NULL. Le nouvel élément est retourné par la fonction.
3. Écrivez une fonction `trierListe()` qui prend en paramètre un pointeur vers une liste chaînée non triée (par exemple, celle que vous avez construite dans l'exercice 1 avec la fonction `enlisterDebut()`) et copie tous ses éléments dans une nouvelle liste triée par prix croissants.