Université de Valenciennes et du Hainaut-Cambrésis
ENSIAME 2017-2018

Fiche d'Exercices Pratiques 2 : Liste Chaînée

Exercice 1 :

1. Définir un type ListeInt, liste chaînée d'entiers.
2. Écrire un sous-programme de création d'une ListeInt contenant un élément (saisi au clavier).
3. Écrire un sous-programme d'ajout en tête de liste d'un entier donné dans une ListeInt.
4. Écrire un sous-programme d'ajout en queue d'un entier donné dans une ListeInt.
5. Écrire un sous-programme d'affichage des éléments d'une ListeInt donnée.
6. Écrire un sous-programme de libération de la mémoire d'une ListeInt donnée.
7. Écrire un sous-programme d'ajout d'un entier donnée à sa place si la liste en entrée est supposée triée dans l'ordre croissant.

Exercice 2 :

On considère une structure Voiture composée des champs id (un identifiant entier), mm (une marque et un modèle, une chaîne de caractères), an (une année), cout (un coût réel) et km (un kilométrage (entier)).

1. Après avoir défini le type Voiture, donner un type LVoiture, liste chaînée de voitures.
2. Écrire un sous-programme de saisie d'une Voiture, puis un sous-programme de création d'une
3. LVoiture.
4. Écrire un sous-programme d'affichage (à l'endroit) d'une LVoiture donnée.
5. Écrire un sous-programme qui recherche la première voiture contenue dans une LVoiture donnée ayant son identifiant supérieur à une valeur donnée.
6. Écrire un sous-programme qui ajoute une voiture donnée juste après une voiture connue par son identifiant dans une LVoiture.
7. Écrire un sous-programme de libération de la mémoire d'une LVoiture donnée.

Exercice 3 :

1. Définir un type appelé ListeReel, liste chaînée de réels.
2. Écrire une fonction qui calcule et retourne le nombre d'éléments (i.e., la longueur) d'une ListeReel donnée.
3. Écrire une fonction qui calcule et retourne le nombre d'occurrences d'une valeur r donnée dans une ListeReel donnée.
4. Écrire un sous-programme qui détermine la valeur se trouvant à la position pos donnée dans une ListeReel donnée.
5. Écrire un sous-programme qui supprime l'élément de tête d'une ListeReel donnée.
6. Écrire un sous-programme qui supprime l'élément de queue d'une ListeReel donnée.
7. Écrire un sous-programme qui ajoute la valeur v donnée dans une ListeReel donnée en position pos.
8. Écrire un sous-programme qui, à partir de deux ListeReel supposées triées en ordre croissant, construit et retourne une ListeReel contenant toutes les valeurs, et triée en ordre croissant.
9. Écrire un sous-programme qui construit et retourne l'union de deux ListeReel supposées triées en ordre croissant.
10. Écrire un sous-programme qui supprime tous les éléments dupliqués dans une ListeReel donnée supposée triée (ordre croissant).
11. Écrire un sous-programme qui, à partir d'une ListeReel donnée, construit deux autres listes. La première contient les valeurs positives de la liste initiale, et la seconde les valeurs négatives.

Exercice 4 :

On considère dans un premier temps une liste doublement chaînée d'entiers, avec un pointeur sur la tète et un pointeur sur la queue.

1. Définir un type DListeInt permettant de manipuler une telle liste.
2. Écrire un sous-programme pour ajouter un élément en tète de liste.
3. Écrire un sous-programme pour ajouter un élément en queue de liste.

4. Écrire un sous-programme pour supprimer le dernier élément de la liste.

On considère maintenant une liste chaînée circulaire d'entiers, le dernier élément pointant sur la tète.
5. Définir un type CListeInt pour manipuler cette liste.
6. Écrire un sous-programme qui créer une liste avec un seul élément.
7. Écrire un sous-programme qui ajoute un élément en tête de liste.
8. Écrire un sous-programme qui affiche une CListeInt donnée.
9. Écrire un sous-programme qui libère la mémoire allouée pour une CListeInt.

Finalement, on considère une liste doublement chaînée circulaire.
10. Définir un type DCListeInt pour cette liste.
11. Écrire un sous-programme qui créer un e liste avec un seul élément.
12. Écrire un sous-programme qui ajoute un élément en queue de liste.
13. Écrire un sous-programme qui libère la mémoire occupée par une DCListeInt donnée.

Exercice 5 :

Dans certaines applications il peut être utile d'utiliser un tableau de listes chaînées. On peut par exemple représenter une matrice creuse (contenant peu de valeurs non nulles) pour éviter d'allouer une grande quantité de mémoire. Ici on suppose vouloir stocker dans un tableau l'ensemble des noms des clients de chaque agent d'une banque. On peut alors associer une liste de noms à chaque agent, et regrouper toutes ces listes dans un tableau de listes, comme illustré ci-dessous :
1. Définir un type ListeNom, liste chaînée de noms (chaque nom ayant au plus 30 caractères).
2. Définir un type TListes, tableau de listes de noms.
3. Écrire un sous-programme qui alloue dynamiquement un tableau de type TListes pouvant contenir n listes, et initialise chacune d'elle à NULL.
4. Écrire un sous-programme qui ajoute un nom donné en tête de la liste d'indice i du tableau.
5. Écrire un sous-programme qui affiche tous les noms de la liste d'indice i du tableau.
6. Écrire un sous-programme qui calcule le nombre total de noms contenus dans le tableau.