

TD les listes chaînées

Exercice 1

On se propose d'implémenter les fonctions de base d'une liste chaînée, sachant qu'elle se représente comme suit :

Structure nœud

Val : entier

Suivant : \uparrow nœud

Fin structure


Ecrire un algorithme qui fait appel aux sous-modules suivants:

1. CREER qui permet de créer une liste chaînée simple d'entiers
2. CONSTRUIRE qui permet de remplir une liste chaînée de plusieurs éléments selon le choix de l'utilisateur
1. EST_VIDE qui permet de dire si une liste est vide ou pas
3. LONGUEUR qui permet de renvoyer le nombre d'éléments dans une liste
4. AFFICHAGE qui permet de parcourir les éléments d'une liste chaînée avec affichage des valeurs
5. DERNIER qui permet de renvoyer le contenu du dernier élément de la liste
6. PREMIER qui renvoie le contenu du premier élément de la liste
7. VIDER qui permet de vider le contenu d'une liste
8. APPARTENIR qui permet d'indiquer si un élément appartient à une liste ou pas
9. CHERCHER qui permet de vérifier si une valeur donnée existe ou pas dans la liste et renvoie sa position
10. OCCUR qui compte le nombre d'occurrences d'une valeur donnée dans une liste chaînée
11. INSERER qui permet d'insérer un nouvel élément
 - a. en tête de liste chaînée simple
 - b. en fin de liste chaînée simple
 - c. avant un élément courant d'une liste chaînée simple
 - d. après un élément courant d'une liste chaînée simple
12. SUPPRIMER qui permet de supprimer un élément d'une liste chaînée simple
 - a. en tête de liste,
 - b. en fin de liste,
 - c. l'élément courant
13. TRANSFERT qui permet de transférer des éléments d'un tableau vers une liste et d'afficher cette liste

Exercice 2

La manipulation d'une liste peut se faire grâce aux opérations suivantes :

- **ListeVide () : Liste**
- **Cons (t : entier, q : Liste) : Liste**
- **Tete (list : Liste) : Entier**
- **Queue (list : Liste) : Liste**

Exemple : 

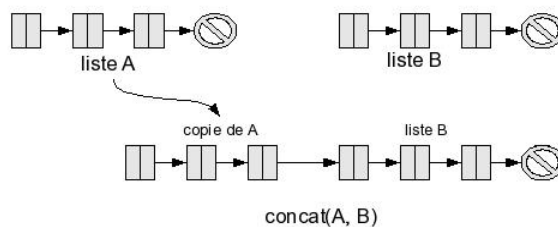
$L = \text{Cons}(4, \text{Cons}(17, \text{Cons}(21, \text{Cons}(13, \text{Cons}(5, \text{Liste Vide}())))))$

En utilisant les fonctions prédéfinies ci-dessus, faites les exercices suivants :

2. Ecrire un algorithme qui permet de construire la liste ci-dessus.
3. Ecrire un sous-programme qui permet d'ajouter une nouvelle valeur entière à une liste de type Element ?
4. Ecrire un sous-programme qui permet de retirer la dernière valeur entière d'une liste de type Element ?
5. Ecrire un sous-programme qui permet de renvoyer le nième élément d'une liste chaînée de type Element ?

Exercice 3

Ecrire un sous-programme qui permet de concaténer deux listes



Exercice 4

Le système informatique d'une agence de location de véhicules permet de gérer les ressources de l'agence. En particulier, le système est capable de stocker les informations relatives aux véhicules en exploitation. On se propose d'écrire un algorithme qui permet la saisie des véhicules.

1. Ecrire un algorithme, qui s'appelle **Agence_Location**, qui permet de définir -avec une structure de données- un nouveau type **Vehicule** comprenant les informations suivantes :
 - l'immatriculation du véhicule ;
 - le type du véhicule ;
 - la marque du véhicule ;
 - la couleur du véhicule.

L'algorithme doit aussi permettre à l'utilisateur de saisir vingt véhicules dans une liste chaînée.

2. En réalité, le nombre de véhicules à saisir dans la liste n'est pas connu. C'est à l'utilisateur de définir ce nombre à chaque utilisation de l'algorithme. D'ailleurs, si on connaissait *a priori* le nombre de véhicules à saisir, on aurait pu dans ce cas utiliser un tableau. Modifier l'algorithme précédent afin que l'utilisateur puisse saisir autant de véhicules qu'il souhaite.

3. On se rend compte que la liste chaînée n'est pas une structure adéquate pour la sauvegarde des véhicules. On se propose donc de saisir les véhicules dans un fichier. Compléter l'algorithme précédent afin de pouvoir écrire le contenu de la liste saisie dans un fichier existant, qui s'appelle "**Vehicules.txt**". Les enregistrements à insérer dans le fichier sont à champs de largeurs fixes.

TD - les piles et les files

Exercice 1

En utilisant une liste chaînée, définir les différentes fonctions et procédures de base d'une pile, sans utiliser les fonctions prédéfinies :

1. CréerPile qui permet d'initialiser la pile.
2. Empile qui permet d'ajouter une valeur val dans la pile.
3. Dépile qui dépile la pile en retournant la valeur du sommet dans val.
4. EstVide qui teste si la pile est vide ou non, et retourne un booléen.
5. Taille qui retourne le nombre d'éléments dans la pile.
6. Sommet qui retourne le sommet de la pile.
7. ViderPile qui permet de vider la pile.

Exercice 2

Ecrire une fonction capable de comparer le contenu d'une liste chaînée et le contenu d'une pile. Dans le cas où la pile contient de son sommet vers sa base, les mêmes éléments que la liste de son début vers sa fin, la fonction doit retourner Vrai. Dans le cas contraire elle retourne Faux.

Proposer deux versions : une version itérative et une version récursive.

Exercice 3

En utilisant une liste chaînée, définir les différentes fonctions et procédures de base d'une file, sans utiliser les fonctions prédéfinies :

1. CréerFile qui crée une file vide.
2. Premier qui retourne le premier élément de la file.
3. Dernier qui retourne le dernier élément de la file.
4. ViderFile qui permet de vider une file.
5. EstVide qui teste si la file est vide et retourne un booléen.
6. Taille qui retourne le nombre d'éléments dans la file.
7. Enfiler qui permet d'ajouter un élément dans la file.
8. Défiler qui permet de retirer un élément de la file et le renvoyer