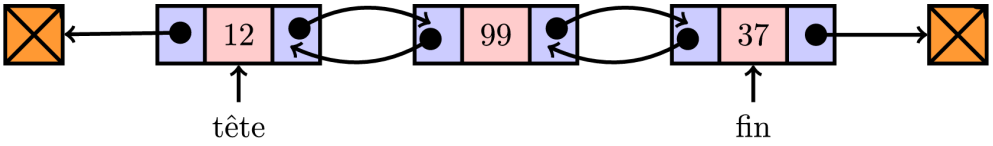
TD2 : Listes doublement chaînées

Exercice 1

Les listes simplement chaînées permettent d'ajouter ou de retirer efficacement en tête, mais la suppression d'un maillon situé ailleurs nécessite la connaissance de son prédécesseur, ce qui impose un parcours depuis le début.

La liste doublement chaînée contourne cette limite en ajoutant à chaque maillon une référence vers son successeur et son prédécesseur. Elle permet un parcours dans les deux sens et simplifie les insertions et suppressions en tout point de la liste.



On souhaite implémenter une telle structure. Chaque maillon contient une donnée entière, une référence vers le maillon précédent et une référence vers le maillon suivant. La liste conserve une référence vers le premier et le dernier maillon. À la création, la liste est vide.

Q1. Définir la classe `Maillon` avec un champ entier `donnee`, une référence `precedent` et une référence `suivant`.

Q2. Définir la classe `ListeDoublementChainee` contenant une référence vers le premier maillon (`tete`) et une référence vers le dernier maillon (`fin`), initialement vides.

Q3. Écrire la méthode `boolean estVide()` qui retourne `true` si la liste est vide.

Q4. Écrire la méthode `void afficher()` qui affiche les éléments de la liste.

Q5. Écrire la méthode `void ajoutDebut(int e)` qui ajoute un élément en tête.

Q6. Écrire la méthode `void ajoutFin(int e)` qui ajoute un élément en fin.

Q7. Écrire la méthode `void retirerDebut()` qui supprime le premier maillon.

Q8. Écrire la méthode `void retirerFin()` qui supprime le dernier maillon.

Q9. Écrire la méthode `Maillon recherche(int e)` qui retourne le premier maillon contenant `e`, ou `null` si absent.

Q10. Écrire la méthode `Maillon accesMaillon(int i)` qui retourne le i-ème maillon, ou `null` si inexistant.

Q11. Écrire la méthode `Maillon insertion(int e, Maillon m)` qui insère un nouveau maillon contenant `e` avant le maillon `m`. Si `m == null`, l'insertion se fait en fin. La méthode retourne le nouveau maillon.

Exercice 2

Écrire une méthode `boolean estPalindrome()` dans la classe `ListeDoublementChainee` qui teste si la liste est un palindrome.

Une liste est dite palindrome si elle se lit de la même manière de gauche à droite et de droite à gauche. Par exemple, la liste `1 → 2 → 3 → 2 → 1` est un palindrome.