Dossier Algorithmique - INF 231

# Introduction

Ce dossier présente différents algorithmes de gestion de listes en langage C, dans le cadre du cours INF 231. Les structures étudiées sont : les listes simplement chaînées, les listes doublement chaînées, ainsi que leurs versions circulaires. L’objectif est de comprendre les opérations classiques d’insertion et de suppression.

# Objectifs

Les exercices portent sur les opérations suivantes :

1. Lire un élément et supprimer toutes ses occurrences dans une liste simplement chaînée.

2. Insertion d’un élément dans une liste simplement chaînée triée.

3. Insertion d’un élément dans une liste doublement chaînée triée.

4. Insertion en tête et en queue dans une liste simplement chaînée circulaire.

5. Insertion en tête et en queue dans une liste doublement chaînée circulaire.

# Algorithmes

## 1. Suppression des occurrences dans une liste simple

Pseudo-code :  
Entrée : liste L, valeur x  
1. Parcourir la liste avec deux pointeurs (courant et précédent).  
2. Si courant->data = x :  
 - Si précédent est NULL, déplacer la tête.  
 - Sinon, relier précédent->next à courant->next.  
3. Libérer le nœud courant.  
4. Continuer jusqu’à la fin de la liste.

## 2. Insertion triée dans une liste simplement chaînée

Pseudo-code :  
Entrée : liste L, valeur x  
1. Créer un nouveau nœud N.  
2. Si la liste est vide ou si x ≤ tête->data : insérer N en tête.  
3. Sinon, avancer dans la liste tant que next->data < x.  
4. Insérer N à la bonne position.

## 3. Insertion triée dans une liste doublement chaînée

Pseudo-code :  
Entrée : liste L, valeur x  
1. Créer un nouveau nœud N.  
2. Si la liste est vide ou x ≤ tête->data : insérer N en tête.  
3. Sinon, avancer jusqu’à trouver la bonne position.  
4. Relier N avec les pointeurs prev et next.  
5. Mettre à jour les liens des nœuds voisins.

## 4. Insertion en tête/queue dans une liste circulaire simple

Pseudo-code pour insérer en tête :  
1. Créer un nouveau nœud N.  
2. Si liste vide : N->next = N.  
3. Sinon, parcourir jusqu’au dernier élément.  
4. Relier N à la tête et mettre à jour le dernier->next.  
5. N devient la nouvelle tête.  
  
Pseudo-code pour insérer en queue :  
1. Créer un nouveau nœud N.  
2. Parcourir jusqu’au dernier élément.  
3. Relier dernier->next à N et N->next à tête.

## 5. Insertion en tête/queue dans une liste circulaire double

Pseudo-code pour insérer en tête :  
1. Créer un nouveau nœud N.  
2. Si liste vide : N->next = N, N->prev = N.  
3. Sinon : mettre à jour les liens entre N, la tête et la queue.  
4. Déplacer la tête sur N.  
  
Pseudo-code pour insérer en queue :  
1. Même principe, mais on insère N après la queue.  
2. Mettre à jour les liens prev/next correctement.

# Analyse de Complexité

1. Suppression des occurrences : O(n)  
2. Insertion triée dans liste simple : O(n)  
3. Insertion triée dans liste double : O(n)  
4. Insertion tête/queue circulaire simple : O(n) pour la queue, O(1) pour la tête  
5. Insertion tête/queue circulaire double : O(1) (car accès direct aux deux extrémités)

# Annexe : Code C complet

Le code source complet est fourni dans le fichier INF231\_listes.c.

# Conclusion

Ces exercices illustrent l’importance des listes chaînées dans la gestion dynamique des données. Chaque variante (simple, double, circulaire) offre des avantages selon les besoins en insertion, suppression et parcours. La maîtrise de ces structures est essentielle pour comprendre les algorithmes avancés.