TP5 : Gestion de la mémoire en C++

Objectifs du TP

À la fin de ce TP, l'étudiant devra être capable de :

- Comprendre la différence entre mémoire **statique** et **dynamique**
- Utiliser les opérateurs new et delete
- Manipuler des **pointeurs**
- Gérer correctement la mémoire pour éviter :
  - Fuites mémoire (memory leaks)
  - Double libération
  - Pointeurs pendants (dangling pointers)
- Implémenter le constructeur, destructeur et constructeur de copie

Contexte

On souhaite développer une application simple de gestion d'un tableau dynamique d'entiers, puis l'améliorer pour gérer une classe contenant une zone mémoire dynamique.

Partie 1 : Allocation dynamique simple

Exercice 1 : Tableau dynamique

Écrire un programme qui :

1. Demande à l'utilisateur la taille n d'un tableau.
2. Alloue dynamiquement un tableau d'entiers.
3. Permet de :
   - Remplir le tableau
   - Afficher son contenu
   - Calculer la somme des éléments
4. Libère correctement la mémoire avant la fin du programme.

Contraintes :

- Utiliser new pour l'allocation.
- Utiliser delete[] pour la libération.
- Vérifier que la taille est strictement positive.

Partie 2 : Gestion mémoire dans une classe

Exercice 2 : Classe TableauDynamique

Créer une classe TableauDynamique contenant :

Attributs :

- int* data
- int taille

Méthodes :

- Constructeur avec taille en paramètre
- Méthode remplir()
- Méthode afficher()
- Méthode setValeur(int index, int valeur)
- Méthode getValeur(int index)

Important :

- Implémenter un **destructeur** qui libère la mémoire.

Partie 3 : Problème de copie (règle des trois)

Exercice 3 : Constructeur de copie

Modifier la classe précédente pour :

1. Ajouter un **constructeur de copie**
2. Ajouter un **opérateur d'affectation**
3. Tester le comportement suivant :

TableauDynamique t1(5);
TableauDynamique t2 = t1;

Question :

Que se passe-t-il si vous ne redéfinissez pas le constructeur de copie ?

Partie 4 : Gestion mémoire avancée

Exercice 4 : Classe Etudiant

Créer une classe Etudiant contenant :

Attributs :

- char* nom (alloué dynamiquement)
- int age

Travail demandé :

1. Constructeur avec paramètres
2. Destructeur
3. Constructeur de copie
4. Opérateur d'affectation
5. Affichage des informations

Éviter :

- Double libération mémoire
- Copie superficielle (shallow copy)

 Questions

1. Quelle est la différence entre :
    - Allocation sur la pile (stack)
    - Allocation sur le tas (heap)
2. Que se passe-t-il si on oublie delete ?
3. Quelle est la différence entre :
    - delete
    - delete[]
4. Expliquer la règle des trois.
5. Que signifie une fuite mémoire ?

1. Remplacer les pointeurs classiques par :
    - std::unique_ptr
    - std::shared_ptr
2. Comparer les comportements.
3. Expliquer l'avantage des smart pointers.