Université Abdelmalek Essaâdi
Faculté des Sciences et Techniques – Tanger
Année Universitaire : 2025/2026
Module : POO en C++, Licence IDAI
Prof. Ikram BEN ABDEL OUAHAB

ATELIER 3

Exercice 1

Une pile est un ensemble dynamique d'éléments où le retrait se fait d'une façon particulière. En effet, lorsque l'on désire enlever un élément de l'ensemble, ce sera toujours le dernier inséré qui sera retiré. Un objet pile doit répondre aux fonctions suivantes :

- Initialiser une pile (constructeur(s))
- Empiler un élément sur la pile (push)
- Dépiler un élément de la pile (pop)

Pour simplifier, nous allons supposer que les éléments à empiler sont de type int.
Le programme principale main comprend la définition d'une classe pile et un programme de test qui crée deux piles p1 et p2, empile dessus des valeurs entières et les dépiler pour vérifier les opérations push et pop.

Exercice 2

Imaginons une application qui traite des fichiers. Ces fichiers vont être lus en mémoire, traités puis sauvegardés. Une fois lu en mémoire, un fichier a deux caractéristiques, une adresse à partir de laquelle se situe le fichier et une longueur, ce qui se concrétisera par un pointeur et une longueur en nombre d'octets.
Imaginons la classe "Fichier" avec un constructeur et un destructeur et les trois méthodes suivantes:

- la méthode "Creation" qui va allouer un certain espace à partir du pointeur P,
- la méthode "Remplit" qui va remplir arbitrairement cet espace (ces remplissages arbitraires sont la preuve de la bonne gestion mémoire car l'accès à une zone non déclarée provoque une violation d'accès),
- la méthode "Affiche" qui va afficher la zone mémoire pointée par P.

Puis écrivons un programme main qui instancie notre classe par new, appelle nos trois méthodes et détruit l'objet par delete et par un destructeur.

Exercice 3

Créez une classe liste simplement chaînée, avec une classe liste. Cette classe a un pointeur sur le premier élément de la liste. Elle a une méthode pour ajouter ou supprimer un élément au début de la liste et une pour afficher la liste en entier.
Evitez toute fuite mémoire. Les éléments de la liste seront contenus dans la structure element.

Exercice 4

On souhaite créer un petit programme de gestion simplifiée de comptes bancaires. Chaque compte appartient à un client et possède un solde. Le programme doit permettre de : *créer des clients et des comptes, copier des comptes, afficher les informations des comptes et des clients, suivre le nombre total de comptes existants, calculer des intérêts sur les comptes.*

1. Classe Client

Créer une classe Client contenant les attributs : un identifiant (int id), un nom (string nom), un prénom (string prenom).

1. Définir les constructeurs nécessaires pour permettre :
   - la création d'un client vide,
   - la création d'un client avec des valeurs initiales,
   - la copie d'un client existant.
2. Ajouter un **destructeur** adapté.
3. Créer une méthode pour afficher les informations d'un client.

Attention : faut-il une copie profonde ou superficielle ? Le destructeur est-il indispensable ?

2. Classe Compte

Créer une classe Compte contenant : un numéro de compte (int numero), un solde (float solde), un **pointeur vers un objet Client** (Client* client), un **attribut partagé** entre tous les comptes indiquant le **nombre total de comptes ouverts**.

1. Ajouter les constructeurs nécessaires pour initialiser correctement un compte :
   - sans paramètre,

- avec paramètres (numéro, solde, client associé),
- par copie.

2. Implémenter le **destructeur** qui libère correctement la mémoire du client associé.
3. Créer une méthode d'affichage des informations du compte (numéro, solde, client).
4. Ajouter une **variable statique** pour compter le nombre total de comptes créés.
   - Incrémenter ce compteur dans les constructeurs.
   - Le décrémenter dans le destructeur.
5. Ajouter une **méthode statique** permettant d'afficher le nombre total de comptes existants.

3. Fonction utilitaire

Créer une fonction globale ***calculInteret(float solde, float taux)*** qui retourne le solde après application des intérêts. *Décider si cette fonction doit être déclarée inline ou non. Justifier votre choix.*

4. Programme principal (main)

Dans la fonction main() :

1. Créer plusieurs clients et comptes.
2. Copier certains comptes pour tester le constructeur de copie.
3. Supprimer certains comptes et observer le comportement du destructeur.
4. Afficher à différentes étapes le nombre total de comptes.
5. Appliquer des intérêts sur certains comptes à l'aide de la fonction calculInteret().

Questions de réflexion :

1. Quelle est la différence entre une **copie superficielle** et une **copie profonde** dans ce contexte ?
2. Pourquoi le compteur du nombre de comptes doit-il être **statique** ?
3. Quelle est la différence entre une **méthode statique** et une **méthode normale** ?
4. Dans quel cas est-il pertinent de rendre une fonction **inline** ?
5. Que se passe-t-il si on oublie de libérer la mémoire dans le destructeur ?