Atelier 4 : Classes, Objets, Constructeurs

Exercice 1

Effectuer les opérations arithmétiques sur des **nombres complexes** à l'aide d'une classe et d'un objet.
Le programme doit demander la partie réelle et imaginaire de deux nombres complexes et afficher les parties réelle et imaginaire de l'opération demandée. (égalité, addition, soustraction, multiplication, division). *Le choix de l'opération peut être fait par un Menu.*

Exercice 2

Ecrire un programme en C++ avec une classe mère **Animal**. À l'intérieur, définir des variables nom et d'âge, et la fonction set_value(). Créer ensuite deux sous classes de base **Zebra** et **Dolphin** qui écrivent un message indiquant l'âge, le nom et donnant des informations supplémentaires (par exemple, le lieu d'origine), Créer 2 variables un de type Zebra et l'autre Dolphin puis appeler la méthode set_value() pour chaque instance.

Exercice 3

Créer une classe **Personne** qui comporte trois champs privés, nom, prénom et date de naissance. Cette classe comporte un constructeur pour permettre d'initialiser des données. Elle comporte également une méthode polymorphe *Afficher* pour afficher les données de chaque personne.

- Créer une classe **Employe** qui dérive de la classe Personne, avec en plus un champ Salaire accompagné de sa propriété, un constructeur et la redéfinition de la méthode Afficher.
- Créer une classe **Chef** qui dérive de la classe Employé, avec en plus un champ Service accompagné de sa propriété, un constructeur et la redéfinition de la méthode Afficher.
- Créer une classe **Directeur** qui dérive de la classe Chef, avec en plus un champ Société accompagné de sa propriété, un constructeur et la redéfinition de la méthode Afficher.

Exercice 4

Réaliser une classe C++ "**vecteur3d**" permettant de manipuler des vecteurs à 3 composantes (de type float). On y prévoira :

- un constructeur, avec des valeurs par défaut (0),
- une fonction d'affichage des 3 composantes du vecteur, sous la forme : (x, y, z)
- une fonction permettant d'obtenir la **somme** de 2 vecteurs ;
- une fonction permettant d'obtenir le **produit** scalaire de 2 vecteurs.
- une fonction **coincide** permettant de savoir si 2 vecteurs ont mêmes composantes.
- une fonction qui renvoie la **norme** du vecteur
- une fonction nommée normax permettant d'obtenir, parmi deux vecteurs, celui qui a la plus grande norme.

On prévoira trois situations :

- le résultat est renvoyé par valeur ;
- le résultat est renvoyé par adresse, l'argument étant également transmis par adresse.
- le résultat est renvoyé par référence, l'argument étant également transmis par référence.

Exercice 5

Ecrire un programme en C++ qui vérifie combien de fois une fonction « call » d'une classe Test a été appelée à partir du programme principal, main.
Note : penser à utiliser une variable static.

Exercice 6

Réaliser une classe point permettant de manipuler un point d'un plan. On prévoira :

- un constructeur recevant en arguments les coordonnées (float) d'un point ;
- une fonction membre ***deplace*** effectuant une translation définie par ses deux arguments (float);
- une fonction membre ***affiche*** se contentant d'afficher les coordonnées cartésiennes du point.

Les coordonnées du point seront des membres donnés privés.
On écrira séparément :

- un fichier source constituant la déclaration de la classe ;
- un fichier source correspondant à sa définition.

Dans le programme principale (main), testez la classe en : déclarant un point, l'affichant, le déplaçant et l'affichant à nouveau.

Exercice 7 (Extrait Examen 2022-2023)

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 8

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 9

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 contenu dans la structure *element*.

Exercice 10

Soit une chaîne de caractères contenant une **date** (JJ/MM/AAAA) et une **heure** (HH:NN) sous la forme JJMMAAAAHHNN.

Par exemple 010920091123 représente la date du 1er septembre 2009 à 11h23.

Créer un programme permettant d'extraire les différents champs et de les afficher.

Exercice 11 (Extrait examen 2023-2024)

Écrire une classe **Traitement** en C++ qui implémente les méthodes suivantes :

1. La méthode « **Initialise** » qui demande à l'utilisateur de saisir 15 entiers, un par un, puis stoke ces entiers dans un vecteur (**attribut de la classe**), le programme vérifie les valeurs saisies par l'utilisateur (**cas de chaine de caractère par exemple**), la méthode doit uniquement accepter des entiers **pairs** et n'accepte pas **les valeurs nulles « 0 »**.
2. La méthode **show** qui affiche les éléments de vecteur **en utilisant la récursivité**.
3. Les deux méthodes **amies** qui renvoient un **double**, la première « **moyenne** » pour calculer la moyenne du vecteur et la deuxième « **médian** » pour calculer la médian de vecteur.