TD/TP03

Exercice 01

Créez une classe nommée **Rectangle** pour représenter un rectangle. Cette classe doit inclure deux variables membres pour stocker la longueur et la largeur, ainsi qu'une méthode permettant de calculer et de retourner la surface du rectangle.

Exercice 02

Concevez une classe **Somme** pour effectuer l'addition de deux nombres. Cette classe doit contenir deux variables membres pour stocker les nombres et une méthode qui calcule leur somme. Dans le programme principal (`main`), demandez à l'utilisateur de fournir deux nombres entiers, passez-les au constructeur de la classe, et affichez le résultat de leur addition.

Exercice 03

Créez une classe **Student** pour gérer les informations d'un étudiant. Cette classe doit contenir les données suivantes : le nom de l'étudiant (type `char`) et deux notes (type `double`). Implémentez une méthode pour calculer la moyenne des deux notes et une autre méthode pour afficher le nom de l'étudiant ainsi que sa moyenne. Le programme principal doit demander à l'utilisateur d'entrer le nom et les notes de l'étudiant, puis afficher ses informations ainsi que sa moyenne.

Exercice 04

Créez une classe **Complex** pour modéliser des nombres complexes et effectuer des opérations arithmétiques à l'aide de fonctions membres et de fonctions amies. L'objectif est de surcharger les opérateurs arithmétiques afin de simplifier l'utilisation de la classe pour des calculs complexes.

Spécifications de la classe :

1. **Attributs** :
   - Re : la partie réelle, de type double.
   - Img : la partie imaginaire, de type double.
2. **Méthodes** :
   - Un **constructeur** avec des arguments pour initialiser les parties réelle et imaginaire (avec des valeurs par défaut).
   - Une méthode pour **afficher** un nombre complexe sous une forme lisible.
   - Une méthode qui calcule le **module** ($|z| = \sqrt{Re^2 + Img^2}$).
   - Une méthode qui retourne le **conjugué** ($\bar{z} = Re - i.Img$).
2. **Surcharge des opérateurs** :
   - **Addition** (+) : permet d'ajouter deux nombres complexes ou un nombre complexe avec un scalaire.
   - **Soustraction** (-) : permet de soustraire deux nombres complexes ou un scalaire.
   - **Multiplication** (*) : permet de multiplier deux nombres complexes.
   - **Division** (/) : permet de diviser deux nombres complexes.

Programme principal :

- Implémentez un programme de test dans la fonction main().
- Demandez à l'utilisateur d'entrer les parties réelle et imaginaire de deux nombres complexes.
- Effectuez les opérations suivantes :
  1. Affichez le module et le conjugué de chaque nombre complexe.
  2. Réalisez des calculs (addition, soustraction, multiplication et division) entre deux nombres complexes ou avec un scalaire.
  3. Affichez les résultats.

TD/TP03

Exercice 05

Créez une classe **Point** pour représenter un point dans un plan cartésien en utilisant des coordonnées x et y.

Spécifications de la classe :

1. **Attributs** :
   - Deux membres privés : x et y (type int) pour stocker les coordonnées du point.
2. **Méthodes** :
   - **Constructeur** : Initialise les coordonnées x et y à l'aide de paramètres, tout en affichant un message indiquant la création du point et son adresse en mémoire.
   - **Destructeur** : Affiche un message indiquant la destruction du point et son adresse en mémoire.
   - **Méthode distance** : Calcule et renvoie la distance entre le point courant et un autre point donné en paramètre, selon la formule :

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Programme principal :

1. Créez un point a avec des coordonnées spécifiques.
2. Copiez ce point dans un autre point b.
3. Affichez la distance entre ces deux points en utilisant la méthode distance.
4. Observez l'exécution du constructeur et du destructeur en notant les messages affichés.

Exercices 06

Créez une classe Tableau qui représente un tableau dynamique avec les fonctionnalités suivantes :

Spécifications de la classe :

1. **Attributs :**
   - **taille :** un entier représentant la taille du tableau.
   - **elements :** un pointeur vers un tableau alloué dynamiquement (de type float).
2. **Méthodes :**
   - **Constructeur avec paramètre :** Initialise un tableau de taille donnée et alloue dynamiquement la mémoire correspondante.
   - **Destructeur :** Libère la mémoire allouée dynamiquement pour éviter les fuites mémoire.
   - **Méthode saisir() :** Permet de remplir le tableau à partir des données entrées par l'utilisateur.
   - **Méthode afficher() :** Affiche le contenu du tableau à l'écran.
   - **Constructeur par copie :** Crée un nouvel objet en copiant les données d'un tableau existant (évite les problèmes liés au partage de pointeurs).
   - **Méthode opposé() :** Retourne un tableau contenant les valeurs opposées de l'objet actuel.

Programme principal :

1. Créez un tableau de taille spécifiée par l'utilisateur.
2. Remplissez le tableau à l'aide de la méthode saisir().
3. Affichez son contenu avec afficher().
4. Copiez le tableau en utilisant le constructeur par copie et affichez son contenu pour vérifier que les données ont été correctement copiées.
5. Utilisez la méthode opposé() pour générer un nouveau tableau avec les valeurs opposées et affichez le résultat.
6. Vérifiez que la mémoire allouée dynamiquement est correctement libérée lors de la destruction des objets.