Programmation orientée objet (C++) – ING2-GSI

TP2 : Constructeurs & destructeur

Objectifs

- Définir des constructeurs et le destructeur d'une classe
- Utilisation de la liste d'initialisation
- Définir le constructeur de copie
- Fonction amie

1 – Personne

Exercice 1.1 :

Créer une classe `Personne` représentée par un `nom`, un `prénom` et un `âge`. Le nom et le prénom seront représentés sous forme de tableau de 20 caractères et l'âge par un entier. On veut pouvoir créer des objets de type Personne soit en spécifiant le nom, le prénom et l'âge, soit en ne spécifiant rien, soit en spécifiant un objet Personne préexistant (constructeur de copie).
Définir les constructeurs et destructeur associés. Afin de pouvoir tester la validité de votre classe, implémenter une fonction d'affichage : `void afficher() const`.
Dans un autre fichier. Faire une fonction `main` permettant de tester votre classe. Ce programme de test créera un tableau de Personne avec différentes initialisations, affichera l'ensemble des objets Personne à l'aide de la fonction `afficher()` et détruira le tableau.

Exercice 1.2 :

Modifier la classe Personne en remplaçant les tableaux par des tableaux dynamiques. Modifier les constructeurs afin qu'ils allouent de la mémoire aux tableaux lors de la création d'un objet et le destructeur afin qu'il libère la mémoire allouée.

2 – Vecteur 3D

Exercice 2.1 :

Ecrivez une classe `Vecteur3D` comportant :

- trois données membre de type double x,y,z (privées)
- deux fonctions membres d'affichage :
    - `void affiche()` affichant le vecteur.

    - `void affiche(const char* string)` affichant string avant l'affichage du vecteur.
- deux constructeurs
    - l'un sans argument, initialisant chaque composante à 0.
    - l'autre, avec trois arguments correspondant aux coordonnées du vecteur.

Modifiez ensuite le code pour que les constructeurs soit des fonctions en ligne (`inline`).

Exercice 2.2 :

Rajouter les fonctions suivantes à la classe Vecteur3D :

- Des fonctions permettant d'accéder aux coordonnées d'un vecteur :
    - `int abscisse()` pour x,
    - `int ordonnée()` pour y et
    - `int cote()` pour z.
- Des fonctions permettant de modifier les coordonnées d'un vecteur :
    - `void abscisse(int nouvelle_abscisse)` pour x,
    - `void ordonnée(int nouvelle_ordonnée)` pour y et
    - `void cote(int nouvelle_cote)` pour z.
- `bool coincide(Vecteur3D v)` qui renvoie `true` si v et l'objet courant ont les mêmes coordonnées, `false` sinon.

Exercice 2.3 :

Rajouter les fonctions suivantes à la classe Vecteur3D :

- `double produit_scalaire(Vecteur3D v)` qui calcule le produit scalaire de `v` avec l'objet courant.
- `Vecteur3D somme(Vecteur3D v)` qui calcule le vecteur somme de `v` avec l'objet courant.

Changer la fonction `coincide` pour qu'elle soit symétrique. Utiliser le prototype suivant : `friend int coincide(Vecteur3D v1, Vecteur3D v2)`. Que signifie le mot-clé `friend` ?