SIM201 TP3
Introduction aux objets

Pensez à structurer l'implémentation, à mettre des commentaires et à passer les arguments par référence en les protégeant (const) si besoin. On écrira l'implémentation dans des fichiers séparés.

1) Ecrire une classe `vecteur` ayant pour données membre *privées* la dimension du vecteur : `int dim_` et un tableau de réels : `float * val_` et en tant que fonctions membre :

- un constructeur par défaut : `vecteur()`
- un constructeur avec dimension et une valeur par défaut : `vecteur(int, float x=0)`
- le constructeur par copie ainsi que l'opérateur `=`
- le destructeur
- une fonction `dim()` retournant la dimension
- une fonction `check_dim(...)` vérifiant la consistance des dimensions de deux vecteurs
- une fonction `val(int) const` donnant accès à la i ème valeur (i=1,dim) en lecture et en écriture
- une fonction d'impression du vecteur : `void print() const`
- des fonctions externes à la classe permettant d'effectuer la **somme** (`somme(...)`), la **différence** (`diff(...)`) de deux vecteurs ainsi que le **produit** (`produit(...)`) d'un scalaire et d'un vecteur
- des fonctions externes réalisant le **produit scalaire** de deux vecteurs et le calcul la **norme euclidienne** d'un vecteur.

Ecrire les définitions dans un fichier entête (vecteur.hpp) et l'implémentation des fonctions membre dans le fichier (vecteur.hpp). Afin de tracer le comportement des constructeurs, inclure dans toutes les fonctions une impression indiquant le passage par la fonction (par exemple `cout<<"construteur par défaut"<<endl;`).

2) Réaliser un programme principal validant toutes les fonctionnalités.

Pour ceux qui en veulent plus ...

3) Ecrire dans le même esprit une classe **matrice** (matrice carrée d'ordre n) utilisant un tableau simple **float * val_**, le rangement s'opérant par colonne ($M_{i,j},\ 1 \leq i,j \leq n$ stocké en position $(j-1)n+i-1$). Ecrire les principaux constructeurs (dimension, copie, opérateur =), l'accès à l'élément (i,j) ainsi que des fonctions d'accès renvoyant un pointeur sur le premier élément d'une ligne et sur le premier élément d'une colonne, une fonction d'impression.

4) Ecrire un produit matrice $\times$ vecteur le plus rapide possible (tester avec la fonction **time**). Comparer en particulier, une version fondée sur l'opérateur d'accès (i,j) et celle fondée sur un accès par pointeur et un algorithme adapté aux pointeurs (déplacement séquentiel) : si `p` désigne un pointeur sur un élément du tableau, `p++` incrémente le pointeur de 1 et `p` pointe alors sur l'élément suivant.