L a b o r a t o i r e

Classe : INF1	**Labo no** : 09	Matrices

But

- Utilisation de tableaux « vector »
- Utilisation intense de la librairie « algorithm »

A faire

Développer un programme mettant à disposition les éléments nécessaires à la manipulation de vecteurs et de matrices d'entiers.

Vocabulaire	Vecteur	tableau (vector) à 1 dimension
	Matrice	tableau à 2 dimensions (vector de vector)

Celui-ci doit contenir les sous-programmes suivants :

Operateur <<	Affiche un Vecteur au format [v_1, v_2, …, v_n]
Operateur <<	Affiche une Matrice au format [[..], [..], [..]]
estCarree	retourne un booléen indiquant si la matrice est carrée
maxCol	Retourne la longueur max des vecteurs d'une matrice / !\ sans boucle => utiliser <algorithm>
sommeLigne	Retourne un vecteur contenant la somme des valeurs de chacune des lignes / !\ une seule boucle => utiliser <algorithm>
vectSommeMin	Retour le vecteur d'une matrice dont la somme est valeur est la plus faible. / !\ sans boucle => utiliser <algorithm>
shuffleMatrice	Mélanger les vecteurs d'une matrice sans altérer les vecteurs. / !\ La *seed* du générateur est basée sur l'heure.
sortMatrice	Trier dans l'ordre croissant une matrice fonction de l'élément max d'un vecteur [[1, 7], [9, 2], [4, 2]] => [[4, 2], [1, 7], [9, 2]] / !\ sans boucle => utiliser <algorithm>
sommeDiagDG	rend par un paramètre la somme des valeurs de la diagonale « / ». Retourne « True » si la matrice est carrée.
sommeDiagGD	rend dans un paramètre la somme des valeurs de la diagonale « \ » … retourne « True » si la matrice est carrée.

La surcharge des opérateurs de flux n'est pas encore vue. Vous pouvez faire une fonction « afficher » et/ou lire **Stream extraction**

Stream extraction and insertion

The overloads of `operator>>` *and* `operator<<` *that take a std::istream& or std::ostream& as the left hand argument are known as insertion and extraction operators. Since they take the user-defined type as the right argument (*`b` *in a@b), they must be implemented as non-members.*

```
std::ostream& operator<<(std::ostream& os, const T& obj)
{
    // write obj to stream
    return os;
}
```

```
std::istream& operator>>(std::istream& is, T& obj)
{
    // read obj from stream
    if( /* T could not be constructed */ )
        is.setstate(std::ios::failbit);
    return is;
}
```