```
/*
-----------------------------------------------------------------------------
Nom du fichier : main.cpp
Nom du labo    : Labo 07 : Vecteur et Matrice
Auteur(s)      : Laetitia Guidetti et Dorian Gillioz
Date creation  : 08.12.2021
Description    : Ce programme permet de tester le fonctionnement de la librairie
                 manipulationVecteur.

Remarque(s)    : -

Compilateur    : Mingw-w64 g++ 11.1.0
-----------------------------------------------------------------------------
*/
#include <iostream>                // cout
#include <vector>                  // Utilisation des vecteurs
#include <cstdlib>                 // EXIT_SUCCESS
#include <limits>                  // numeric_limits
#include "manipulationVecteur.h"

using namespace std;

using Vecteur = vector<int>;
using Matrice = vector<Vecteur>;

int main() {

   cout << "Programme montrant le bon fonctionnement de la librairie "
        << "manipulationVecteur" << endl << endl;
   cout << boolalpha;

   //-------------------------------------------------
   // Matrice 1
   //-------------------------------------------------
   Matrice m1 = {{4, 1, 2},
                 {1, 0, 6},
                 {9, 4, 5}};

   cout << "Matrice 1                                : " << m1               << endl;
   cout << "Est carree                               : " << estCarree(m1)    << endl;
   cout << "Est reguliere                            : " << estReguliere(m1) << endl;
   cout << "Longueur minimale des vecteurs           : " << minCol(m1)       << endl;
   cout << "Somme des valeurs de chaque ligne        : " << sommeLigne(m1)   << endl;
   cout << "Somme des valeurs de chaque colonne      : " << sommeColonne(m1) << endl;
   cout << "Vecteur avec la somme la plus faible     : " << vectSommeMin(m1) << endl;
   shuffleMatrice(m1);
   cout << "Melange des vecteurs                     : " << m1               << endl;
   sortMatrice(m1);
   cout << "Tri croissant selon les valeurs minimales : " << m1               << endl;

   cout << "-----------------------------------------------------------" << endl  << endl;

   //-------------------------------------------------
   // Matrice 2
   //-------------------------------------------------
   Matrice m2 = {{5, 7, 2},
                 {1, 12},
                 {7, 0, 1},
                 {3, 5, 1, 8, 2},
                 {2, 0, 8}};

   cout << "Matrice 2                                : " << m2               << endl;
   cout << "Est carree                               : " << estCarree(m2)    << endl;
   cout << "Est reguliere                            : " << estReguliere(m2) << endl;
   cout << "Longueur minimale des vecteurs           : " << minCol(m2)       << endl;
   cout << "Somme des valeurs de chaque ligne        : " << sommeLigne(m2)   << endl;
   cout << "Somme des valeurs de chaque colonne      : " << sommeColonne(m2) << endl;
   cout << "Vecteur avec la somme la plus faible     : " << vectSommeMin(m2) << endl;
   shuffleMatrice(m2);
   cout << "Melange des vecteurs                     : " << m2               << endl;
   sortMatrice(m2);
   cout << "Tri croissant selon les valeurs minimales : " << m2               << endl;
```

```
   cout << "-------------------------------------------------------" << endl  << endl;

   //-------------------------------------------------
   // Quitter le programme
   //-------------------------------------------------
   cout << "Presser ENTER pour quitter";
   cin.ignore(numeric_limits<streamsize>::max(), '\n');
   return EXIT_SUCCESS;
}
```

---

---

```
/*
-----------------------------------------------------------------------------
Nom du fichier : manipulationVecteur.h
Nom du labo    : Labo 07 : Vecteur et Matrice
Auteur(s)      : Laetitia Guidetti et Dorian Gillioz
Date creation  : 08.12.2021
Description    : Déclaration de fonctions permettant de réaliser divers
                 manipulations sur des matrices et vecteurs.
Remarque(s)    : -

Compilateur    : Mingw-w64 g++ 11.1.0
-----------------------------------------------------------------------------
*/

#ifndef LABO07_MANIPULATIONVECTEUR_H
#define LABO07_MANIPULATIONVECTEUR_H

#include <vector>       // Utilisation des vecteurs
#include <iostream>     // ostream

/**
 * Nom             operator
 * But             Permettre de formater l'affichage d'un vecteur
 * @param os       Le flux
 * @param vecteur  Le vecteur à afficher au format
 * @return         Une référence au flux avec le formatage du vecteur
 */
std::ostream& operator<< (std::ostream& os, const std::vector<int>& vecteur);

/**
 * Nom              operator
 * But              Permettre de formater l'affichage d'une matrice (vecteur 2d)
 * @param os        Le flux
 * @param matrice   La matrice à afficher au format
 * @return          Une référence au flux avec le formatage de la matrice
 */
std::ostream& operator<< (std::ostream& os, const std::vector<std::vector<int>>& matrice);

/**
 * Nom              estCarree
 * But              Permettre de déterminer si une matrice est carrée, c'est-à-dire
 *                  le même nombre de lignes que de colonnes et qui est régulière
 * @param matrice   La matrice que l'on veut contrôler
 * @return          true si la matrice est carrée et false si elle ne l'est pas
 */
bool estCarree(const std::vector<std::vector<int>>& matrice);

/**
 * Nom              estReguliere
 * But              Permettre de déterminer si une matrice est régulière,
 *                  c'est-à-dire que toutes les lignes ont la même longueur
 * @param matrice   La matrice que l'on veut contrôler
 * @return          true si elle est régulière et false si elle ne l'est pas
 */
bool estReguliere(const std::vector<std::vector<int>>& matrice);
```

```
/**
 * Nom              minCol
 * But              Déterminer la longueur minimum des vecteurs d'une matrice
 * @param matrice   Matrice dans laquelle le vecteur est cherché
 * @return          La longueur du vecteur le plus petit trouvé
 */
size_t minCol(const std::vector<std::vector<int>>& matrice);

/**
 * Nom              sommeLigne
 * But              Calculer la somme des valeurs de chaque ligne
 * @param matrice   Matrice contenant les lignes à calculer
 * @return          Un vecteur contenant la somme de chaque ligne
 */
std::vector<int> sommeLigne(const std::vector<std::vector<int>>& matrice);

/**
 * Nom              sommeColonne
 * But              Calculer la somme de chaque colonne de la matrice
 * @param matrice   Matrice contenant les lignes à calculer
 * @return          Un vecteur contenant la somme de chaque colonne
 */
std::vector<int> sommeColonne(const std::vector<std::vector<int>>& matrice);

/**
 * Nom              vectSommeMin
 * But              Trouver le vecteur d'une matrice dont la somme des valeurs est
 *                  la plus petite
 * @param matrice   Matrice dans laquelle le vecteur est cherché
 * @return          Vecteur dont la somme est la plus faible
 */
std::vector<int> vectSommeMin(const std::vector<std::vector<int>>& matrice);

/**
 * Nom              shuffleMatrice
 * But              Mélanger les vecteurs de la matrice entre eux sans modifier les vecteurs
 *                  en eux-même
 * @param matrice   La matrice à mélanger
 */
void shuffleMatrice(std::vector<std::vector<int>>& matrice);

/**
 * Nom              sortMatrice
 * But              Trier dans l'ordre croissant une matrice en fonction de
 *                  l'élément minimum de chaque vecteur
 * @param matrice   Matrice à trier
 */
void sortMatrice(std::vector<std::vector<int>>& matrice);


#endif //LABO07_MANIPULATIONVECTEUR_H


 ------------------------------------------------------------------------------------
 ------------------------------------------------------------------------------------


/*
------------------------------------------------------------------------------
Nom du fichier : manipulationVecteur.cpp
Nom du labo    : Labo 07 : Vecteur et Matrice
Auteur(s)      : Laetitia Guidetti et Dorian Gillioz
Date creation  : 08.12.2021
Description    : Définition de fonctions permettant de réaliser divers
                 manipulations sur des matrices et vecteurs.
Remarque(s)    : -

Compilateur    : Mingw-w64 g++ 11.1.0
------------------------------------------------------------------------------
*/

#include <algorithm>            // min_element, max_element, transform, sort, ...
```

```
#include <vector>               // Utilisation des vecteurs
#include <numeric>              // accumulate
#include <chrono>               // chrono::system_clock
#include <random>               // default_random_engine
#include <iostream>             // cout, ostream

#include "manipulationVecteur.h"


using namespace std;

using Vecteur = vector<int>;
using Matrice = vector<Vecteur>;

//-------------------------------------------------
// Déclaration
//-------------------------------------------------

/**
 * Nom              estPasEgale
 * But              Comparer si la taille de deux vecteurs est égale
 * @param vecteur1  Le premier vecteur
 * @param vecteur2  Le deuxième vecteur
 * @return          true s'ils ne sont pas égaux, false sinon
 */
bool estPasEgale(const Vecteur& vecteur1, const Vecteur& vecteur2);

/**
 * Nom              estPlusPetit
 * But              Déterminer le plus petit des vecteurs passé en paramètre
 * @param vecteur1  Le premier vecteur
 * @param vecteur2  Le deuxième vecteur
 * @return          Retourne true si vecteur1 est strictement plus petit que
 *                  vecteur2, false dans le cas contraire
 */
bool estPlusPetit(const Vecteur& vecteur1, const Vecteur& vecteur2);

/**
 * Nom              sommeElement
 * But              Calculer la somme de tous les éléments présents dans un vecteur
 * @param vecteur   Le vecteur contenant les éléments
 * @return          La somme calculée
 */
int sommeElement(const Vecteur& vecteur);

/**
 * Nom              additionValeurs
 * But              Additionner 2 valeurs
 * @param valeur1   La première valeur à additionner
 * @param valeur2   La deuxième valeur à additionner
 * @return          Le résultat de l'addition des deux valeurs
 */
int additionValeurs(int valeur1, int valeur2);

/**
 * Nom              minElement
 * But              Déterminer quel vecteur possède le plus petit élément
 * @param vecteur1  Le premier vecteur
 * @param vecteur2  Le deuxième vecteur
 * @return          true si vecteur1 possède l'élément le plus petit, false dans le
 *                  cas contraire
 */
bool minElement(const Vecteur& vecteur1, const Vecteur& vecteur2);

//-------------------------------------------------
// Définition
//-------------------------------------------------

ostream& operator<< (ostream& os, const Vecteur& vecteur) {
   os << "(";
   for (Vecteur::const_iterator i = vecteur.cbegin(); i != vecteur.cend(); ++i) {
      if (i != vecteur.cbegin()) {
```

```
         os << ", ";
      }
      os << *i;
   }
   os << ")";
   return os;
}

ostream& operator<< (ostream& os, const Matrice& matrice) {
   os << "[";
   for (Matrice::const_iterator i = matrice.cbegin(); i != matrice.cend(); ++i) {
      if (i != matrice.cbegin()) {
         os << ", ";
      }
      os << *i;
   }
   os << "]";
   return os;
}

bool estCarree(const Matrice& matrice) {
   if (matrice.empty()) {
      return true;
   }
   return estReguliere(matrice) && matrice.size() == matrice[0].size();
}

bool estPasEgale(const Vecteur& vecteur1, const Vecteur& vecteur2) {
   return vecteur1.size() != vecteur2.size();
}

bool estReguliere(const Matrice& matrice) {
   return matrice.end() == adjacent_find(matrice.begin(), matrice.end(), estPasEgale);
}

bool estPlusPetit(const Vecteur& vecteur1, const Vecteur& vecteur2) {
   return vecteur1.size() < vecteur2.size();
}

size_t minCol(const Matrice& matrice) {
   if(matrice.empty()) {
      return 0;
   }
   return (*min_element(matrice.cbegin(), matrice.cend(), estPlusPetit)).size();
}

int sommeElement(const Vecteur& vecteur) {
   return accumulate(vecteur.cbegin(), vecteur.cend(), 0);
}

Vecteur sommeLigne(const Matrice& matrice) {
   // Vecteur dont la taille est égale au nombre de ligne de matrice
   Vecteur vecteur(matrice.size());
   transform(matrice.cbegin(), matrice.cend(), vecteur.begin(), sommeElement);
   return vecteur;
}

int additionValeurs(int valeur1, int valeur2) {
   return valeur1 + valeur2;
}

Vecteur sommeColonne(const Matrice& matrice) {
   if(matrice.empty()) {
      return {};
   }
   // La taille du vecteur est égale à la ligne la plus longue de la matrice
   Vecteur vecteurSomme((*max_element(matrice.cbegin(), matrice.cend(),
                                      estPlusPetit)).size());
   // Addition de chaque ligne de matrice dans vecteurSomme
   for(Matrice::const_iterator i = matrice.cbegin(); i != matrice.cend(); ++i) {
      transform((*i).cbegin(), (*i).cend(), vecteurSomme.begin(),
                vecteurSomme.begin(), additionValeurs);
```

```
    }
    return vecteurSomme;
}

Vecteur vectSommeMin(const Matrice& matrice) {
    if(matrice.empty()) {
        return {};
    }
    // Vecteur contenant la somme de chaque ligne de la matrice
    const Vecteur vecteur(sommeLigne(matrice));
    const Vecteur::const_iterator resultat = min_element(vecteur.cbegin(),
                                                         vecteur.cend());

    return matrice[(size_t)distance(vecteur.cbegin(), resultat)];
}

void shuffleMatrice(Matrice& matrice) {
    // http://www.cplusplus.com/reference/algorithm/shuffle/?kw=shuffle
    unsigned seed = chrono::system_clock::now().time_since_epoch().count();
    shuffle(matrice.begin(), matrice.end(), default_random_engine(seed));
}

bool minElement(const Vecteur& vecteur1, const Vecteur& vecteur2) {

    // Vérifie si l'un des vecteurs est vide
    if (vecteur1.empty()) {
        return true;
    }
    if (vecteur2.empty()) {
        return false;
    }

    // Compare la valeur de l'élément le plus petit de chaque vecteur
    return *min_element(vecteur1.cbegin(), vecteur1.cend()) <
           *min_element(vecteur2.cbegin(), vecteur2.cend());
}

void sortMatrice(Matrice& matrice) {
    sort(matrice.begin(), matrice.end(), minElement);
}
```