```cpp
/*
  ------------------------------------------------------------------------
  Fichier     : main.cpp
  Nom du labo : 07 - Matrice
  Auteur(s)   : Mario Amos & Théo Coutaudier
  Date        : 8.12.2021
  But         : Tester la librairie qui met à disposition des utilitaires de
                traitement de vecteur ou matrice

  Compilateur : Mingw-w64 g++ 11.2.0
  ------------------------------------------------------------------------
*/
#include <iostream>  //Pour l'affichage
#include <limits>    //Pour numeric_limits<streamsize>::max()

#include "matrice.h"

using namespace std;

int main() {

   //---------------------------------------
   // Déclaration des vecteur et matrice
   //---------------------------------------

   Vecteur v1 = {1, 2, 3};
   Matrice m1 = {{3,2,1},{8,9,4},{3,5,2}};
   Matrice vide = {};

   //---------------------------------------
   // Utilisation des fonctions d'affichaage.cpp et matriceCaracteristique.cpp
   //---------------------------------------

   cout << "Vecteur      : " << v1 << endl
        << "Matrice      : " << m1 << endl
        << "estCarre     : " << (estCarre(m1)     ? "oui" : "non") << endl
        << "estReguliere : " << (estReguliere(m1) ? "oui" : "non") << endl << endl;

   //---------------------------------------
   // Utilisations des fonctions de calculVecteur.cpp
   //---------------------------------------

   cout << "minCol       : " << minCol(m1)         << endl
        << "sommeLigne   : " << sommeLigne(m1)     << endl
        << "sommeColonne : " << sommeColonne(m1)   << endl
        << "vectSommeMin : " << vectSommeMin(m1)   << endl << endl;

   cout << "Avec une matrice vide :" << endl
        << "minCol       : " << minCol(vide)         << endl
        << "sommeLigne   : " << sommeLigne(vide)   << endl
        << "sommeColonne : " << sommeColonne(vide) << endl
        << "vectSommeMin : " << vectSommeMin(vide) << endl << endl;

   //---------------------------------------
   // Utilisation des fonctions de sortMatrice.cpp
   //---------------------------------------

   shuffleMatrice(m1);
   cout << "Matrice Shuffled  : " << m1 << endl;
   sortMatrice(m1);
   cout << "Matrice Sort      : " << m1;

   //---------------------------------------
   // fin de programme
   //---------------------------------------
   cout << endl;
   cout << "Presser ENTER pour quitter";
   cin.ignore(numeric_limits<streamsize>::max(), '\n');

   return EXIT_SUCCESS;
}

```

```
/*
  ---------------------------------------------------------------------------
  Fichier     : matrice.h
  Auteur(s)   : Mario Amos & Théo Coutaudier
  Date        : 08.12.2021
  But         : Librairie mettant à disposition des utilitaires pour le traitement
                de vecteur ou matrice

  Compilateur : Mingw-w64 g++ 11.2.0
  ---------------------------------------------------------------------------
*/

#ifndef INC_07_MATRICE_MATRICE_H
#define INC_07_MATRICE_MATRICE_H

#include <vector>

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

/**
 * Affiche un vecteur au format (x,x,x)
 * @param os Flux de sortie
 * @param v  Vecteur
 * @return   Flux de sortie
 */
std::ostream& operator<< (std::ostream& os, const Vecteur& v);

/**
 * Affiche une matrice au format [(x,x,x), (x,x,x)]
 * @param os Flux de sortie
 * @param v  Vecteur
 * @return   Flux de sortie
 */
std::ostream& operator<< (std::ostream& os, const Matrice& m);

/**
 * Retourne la longueur minimum des vecteurs d'une matrice
 * @param m Matrice
 * @return  Longueur minimum des vecteurs d'une matrice
 */
unsigned minCol(const Matrice& m);

/**
 * Retourne un vecteur contenant la somme des valeurs de chacune des lignes.
 * @param m Matrice
 * @return Vecteur contenant la somme des valeurs de chacune des lignes
 */
Vecteur sommeLigne(const Matrice& m);

/**
 * Retourne un vecteur contenant la somme des valeurs de chacune des colonnes
 * @param m Matrice
 * @return  vecteur contenant la somme des valeurs de chacune des colonnes
 */
Vecteur sommeColonne(const Matrice& m);

/**
 * Retourne le vecteur d'une matrice dont la somme des valeurs est la plus faible.
 * @param m Matrice
 * @return  Vecteur d'une matrice dont la somme des valeurs est la plus faible
 */
Vecteur vectSommeMin(const Matrice& m);

/**
 * Vérifie si tout les vecteurs d'une matrice ont la même taille.
 * @param m Matrice
 * @return  Un bool indiquant si tout les vecteurs ont la même taille
 */
bool estReguliere(const Matrice& m);

/**
```

```
73    * Vérifie si la matrice estCarré, si la matrice est vide elle est considéré carré.
74    * @param m Matrice
75    * @return  un bool indiquant si la matrice est carré ou non.
76    */
77   bool estCarre(const Matrice& m);
78
79   /**
80    * Melange les vecteurs d'une matrice en fonction d'un clock, ne change pas
81    * l'ordre à l'interieur des vecteurs.
82    * @param m Matrice
83    * @return
84    */
85   void shuffleMatrice(Matrice& m);
86
87   /**
88    * Trie la matrice en fonction du min_element de chaque vecteur
89    * @param m Matrice
90    * @return
91    */
92   void sortMatrice(Matrice& m);
93
94   #endif //INC_07_MATRICE_MATRICE_H
95
```

```cpp
/*
  ------------------------------------------------------------------------
  Fichier     : affichageMatrice.cpp
  Auteur(s)   : Mario Amos & Théo Coutaudier
  Date        : 08.12.2021
  But         : Librairie mettant à disposition des utilitaires pour l'affichage
                de vecteur ou matrice'

  Compilateur : Mingw-w64 g++ 11.2.0
  ------------------------------------------------------------------------
*/

#include <iostream>

#include "matrice.h"

using namespace std;

ostream &operator<<(ostream& os, const Vecteur& v) {
   os << "(";
   //warning conseil d'utiliser auto ingoré
   Vecteur::const_iterator debut = v.begin();
   for (Vecteur::const_iterator i = debut; i != v.end(); ++i) {
      if (i != debut)
         os << ", ";
      os << *i;
   }
   os << ")";
   return os;
}

ostream &operator<<(ostream &os, const Matrice &m) {
   os << "[";
   //warning conseil d'utiliser auto ingoré
   Matrice::const_iterator debut = m.begin();
   for (Matrice::const_iterator i = debut; i != m.end(); ++i) {
      if (i != debut)
         os << ", ";
      os << *i;
   }
   os << "]";
   return os;
}

```

```cpp
/*
  ------------------------------------------------------------------------
  Fichier     : matriceCaracteristique.cpp
  Nom du labo : 07 - Matrice
  Auteur(s)   : Mario Amos & Théo Coutaudier
  Date        : 8.12.2021
  But         : Librairie permettant de determiner si une matrice est régulière ou
                carré

  Compilateur : Mingw-w64 g++ 11.2.0
  ------------------------------------------------------------------------
*/


#include <algorithm>

#include "matrice.h"

using namespace std;

bool sizeOK = true;
size_t taille ;

/**
 * Vérifie que la taille du vecteur correspond avec taille
 * @param m Matrice
 * @return
 */
void isSizeOk(const Vecteur& v)
{
   //met sizeOK false si la taille du vecteur n'est pas la même que taille
   if (v.size() != taille)
      sizeOK = false;
}

bool estReguliere(const Matrice& m) {

   if(m.empty()) //Vérifie si la matrice est vide
      return true;

   //utilise la taille du premier vecteur de la amtrice comme taille de référence
   taille = m[0].size();

   //Vérifie que chaque vecteur de la matrice m ont la même taille
   for_each(m.begin(),m.end(), isSizeOk);
   return sizeOK;
}

bool estCarre(const Matrice& m){
   //Vérifie si la matrice est vide ou régulière
   return m.empty() || (estReguliere(m) && m[0].size() == m.size());
}


```

```cpp
/*
  ---------------------------------------------------------------------------
  Fichier     : calculMatrice.cpp
  Auteur(s)   : Mario Amos & Théo Coutaudier
  Date        : 08.12.2021
  But         : Librairie mettant à disposition des utilitaires pour le traitement
                de vecteur ou matrice

  Compilateur : Mingw-w64 g++ 11.2.0
  ---------------------------------------------------------------------------
*/

#include <algorithm>
#include <numeric>

#include "matrice.h"

using namespace std;

/**
 * @param v1 Premier vecteur à comparer
 * @param v2 Second vecteur à comparer
 * @return Retourne true si v1 est plus petit ou égal que v2, sinon false
 */
bool comparerTailleVecteur(const Vecteur& v1, const Vecteur& v2);

/**
 * Trouve la taille maximum des colonnes d'une matrice
 * @param m
 * @return
 */
unsigned maxCol(const Matrice& m);

/**
 * Effectue la somme de toutes les valeurs d'un vecteur
 * @param v
 * @return
 */
int sommeVecteur(const Vecteur& v);

/**
 * Effectue l'addtion la valeur de deux cellules d'un vecteur
 * @param v1 Valeur 1
 * @param v2 Valeur 2
 * @return Valeurs additionnées
 */
int add(int v1, int v2);

/**
 * Est équivalent à l'addition de deux vecteur:
 * Valeur de retour = {v1[0]+v2[0],v1[1]+v2[1],v1[2]+v2[2],...}
 * @param v1
 * @param v2
 * @return
 */
Vecteur additionColonne(Vecteur v1, Vecteur v2);

unsigned minCol(const Matrice& m) {
   if (m.empty())
      return 0;
   return (*min_element(m.begin(), m.end(), comparerTailleVecteur)).size();
}

Vecteur sommeLigne(const Matrice& m) {
   Vecteur resultat(m.size());
   transform(m.begin(), m.end(), resultat.begin(), sommeVecteur);
   return resultat;
}

Vecteur sommeColonne(const Matrice& m) {
   return accumulate(m.begin(), m.end(), Vecteur(maxCol(m), 0), additionColonne);
}
```

```
73
74  Vecteur vectSommeMin(const Matrice& m) {
75     if (m.empty())
76        return {};
77
78     Vecteur somme = sommeLigne(m);
79     //On récupère un itérateur de min_element, on en soustrait le début du vecteur
80     //pour en ressortir l'index
81     unsigned index = min_element(somme.begin(), somme.end()) - somme.begin();
82     return m[index];
83  }
84
85  bool comparerTailleVecteur(const Vecteur& v1, const Vecteur& v2) {
86     return v1.size() <= v2.size();
87  }
88
89  unsigned maxCol(const Matrice& m) {
90     if (m.empty())
91        return 0;
92     return (*max_element(m.begin(), m.end(), comparerTailleVecteur)).size();
93  }
94
95  int sommeVecteur(const Vecteur& v) {
96     return accumulate(v.begin(), v.end(), 0);
97  }
98
99  int add(int v1, int v2) {
100    return (v1 + v2);
101 }
102
103 Vecteur additionColonne(Vecteur v1, Vecteur v2) {
104    //On doit resize le vecteur le plus petit, sinon cela va aller chercher une
105    //valeur "aléatoire" en mémoire pour l'addition
106    const size_t TAILLE_MAX = max(v1.size(), v2.size());
107    if (v1.size() < TAILLE_MAX)
108       v1.resize(TAILLE_MAX, 0);
109    if (v2.size() < TAILLE_MAX)
110       v2.resize(TAILLE_MAX, 0);
111
112    Vecteur res(TAILLE_MAX, 0);
113    //Additionne des deux vecteurs
114    transform(v1.begin(), v1.end(), v2.begin(), res.begin(), add);
115    return res;
116 }
```

```cpp
/*
  ------------------------------------------------------------------------
  Fichier     : sortMatrice.cpp
  Nom du labo : 07 - Matrice
  Auteur(s)   : Mario Amos & Théo Coutaudier
  Date        : 8.12.2021
  But         : Librairie mettant à disposition des utilitaires pour mélanger ou
                trier une matrice

 Référence    : http://www.cplusplus.com/reference/algorithm/shuffle/?kw=shuffle
                https://www.cplusplus.com/reference/algorithm/sort/

Compilateur   : Mingw-w64 g++ 11.2.0

  ------------------------------------------------------------------------
*/

#include <algorithm>
#include <random>       // std::default_random_engine
#include <chrono>       // std::chrono::system_clock

#include "matrice.h"

using namespace std;


/**
 * Indique quel vecteur possede le plus petit ellement entre v1 et v2
 * @param m Matrice
 * @return 1 si le plus petit element de v1 et plus petit que celui de v2
 */
bool minElement(const Vecteur& v1, const Vecteur& v2);


void shuffleMatrice(Matrice& m)
{
   //creer une seed en fonction du temps
   unsigned seed = chrono::system_clock::now().time_since_epoch().count();
   //warning car conversion long long en unsigned perte de precision
   shuffle(m.begin(), m.end(), default_random_engine(seed)); //mélange aleatoirement
}

void sortMatrice(Matrice& m)
{
   sort(m.begin(), m.end(), minElement); //trie en fonction de minElement
}

bool minElement(const Vecteur& v1, const Vecteur& v2)
{
   return *min_element(v1.begin(), v1.end()) < *min_element(v2.begin(), v2.end());
}
```