```
1
    /*-----
              : main.cpp
: Maëlle Vogel and Tobie Praz
2
3
   Authors
   Creation date : 01.12.2020
                : The program tests each functions created in library.cpp
    Description
 6
                   - display a vector
7
                   - display a matrix
8
                   - check if a matrix is a square
9
                   - check if a matrix is regular
10
                   - return the size of the longest vector
11
                   - return a vector containing the sum of each vector
12
                   - return the vector with the smallest sum
13
                   - shuffle the vector order in the matrix
                   - sort the matrix by the biggest number in a vector
14
15
                   - sum the right to left diagonal /
16
                   - sum the left to right diagonal \
17
    Compiler : Mingw-w64 g++ 8.1.0
18
19
20
    #include <cstdlib>
21
   #include <iostream>
22
    #include <limits>
    #include "library.h"
23
24
25
    using namespace std;
26
27
    #define EMPTY BUFFER cin.ignore(numeric limits<streamsize>::max(), '\n');
28
29
    int main() {
30
31
       //INIT MATRIX
32
       IntMatrix matrix1 = \{\{12, 2, 43\},
33
                         {32, 2, 21}};
34
       IntMatrix matrix2 = \{\{1, 25, 2\},
35
                         {3, 38, 1},
36
                          {4, 23, 1}};
37
       IntMatrix matrix3 = \{\{16, 2, 2\},
                         {3, 34, 17, 5},
38
39
                          {4, 43, 1}};
40
41
       cout << boolalpha;</pre>
42
       cout << "-----" << endl;
       cout << matrix1 << " is square: " << isSquare(matrix1) << endl;</pre>
43
       cout << matrix2 << " is square: " << isSquare(matrix2) << endl;</pre>
44
45
       cout << "----" << endl;
46
       cout << matrix1 << " is regular: " << isRegular(matrix1) << endl;</pre>
47
       cout << matrix2 << " is regular: " << isRegular(matrix2) << endl;</pre>
48
49
       cout << matrix3 << " is regular: " << isRegular(matrix3) << endl;</pre>
50
      cout << "-----" << endl;
51
52
       cout << matrix2 << " max vector size: " << maxCol(matrix2) << endl;</pre>
       cout << matrix3 << " max vector size: " << maxCol(matrix3) << endl;</pre>
53
54
55
       cout << "----" << endl;
56
       cout << matrix2 << " line sum: " << lineSum(matrix2) << endl;</pre>
       cout << matrix3 << " line sum: " << lineSum(matrix3) << endl;</pre>
57
58
59
       cout << "-----" << endl;
       cout << matrix2 << " vector with min line sum: " << vectMinSum(matrix2) << endl;</pre>
60
       cout << matrix3 << " vector with min line sum: " << vectMinSum(matrix3) << endl;</pre>
61
62
63
       cout << "-----" << endl;
       cout << matrix1 << " before shuffle" << endl;</pre>
64
65
       shuffleMatrix(matrix1);
66
       cout << matrix1 << " after shuffle" << endl;</pre>
       cout << matrix2 << " before shuffle" << endl;</pre>
67
68
       shuffleMatrix(matrix2);
69
       cout << matrix2 << " after shuffle" << endl;</pre>
70
71
       cout << "-----" << endl;
72
       cout << matrix1 << " before sort" << endl;</pre>
```

main.cpp M. Vogel & T. Praz – HEIG-VD

```
73
         sortMatrix(matrix1);
        cout << matrix1 << " after sort" << endl;</pre>
74
75
76
         cout << "-----" << endl;
77
         int resultLR1;
78
         cout << matrix2 << " left to right diagonal exists: " << diagLRSum(matrix2, resultLR1)</pre>
79
              << ", result: " << resultLR1 << endl;
80
81
         int resultRL1;
         cout << matrix2 << " right to left diagonal exists: " << diagRLSum(matrix2, resultRL1)</pre>
82
83
              << ", result: " << resultRL1 << endl;
84
85
         int resultLR2;
        \verb|cout| << \verb|matrix1| << \verb|"| left| to right| diagonal exists: "| << \verb|diagLRSum| (matrix1, resultLR2)|
86
             << ", result: " << resultLR2 << endl;
87
88
         cout << endl;</pre>
89
90
        cout << "ENTER FOR EXIT";</pre>
        EMPTY_BUFFER
91
92
         return EXIT_SUCCESS;
93
     }
```

```
1
     #ifndef LABO5 VECTEURS LIBRARY H
     #define LABO5 VECTEURS LIBRARY H
 2
 3
 4
     #include <string>
     #include <vector>
 6
 7
     using namespace std;
 8
 9
     using IntVector = vector<int>;
10
     using IntMatrix = vector<IntVector>;
11
12
     ostream &operator<<(ostream &os, const IntVector &v);</pre>
13
     ostream &operator<<(ostream &os, const IntMatrix &m);</pre>
14
15
      * Is the matrix a square ? (N \times N)
16
      \star @param matrix: the matrix to analyse
17
      * @return True if the matrix is a square,
18
19
                 false if not
20
      * /
21
     bool isSquare(const IntMatrix &matrix);
22
23
      * Is the matrix regular ? (All lines same size)
24
25
      ^{\star} \mbox{\em Qparam} matrix: the matrix to check
26
      * @return True if the matrix is regular,
27
                 false if not
      * /
28
29
     bool isRegular(const IntMatrix &matrix);
30
31
     \,^\star Returns the size of the longest vector of a matrix
32
33
      \star \mbox{\em Qparam} matrix: the matrix to analyse
34
      * @return Longest line size
35
36
     int maxCol(const IntMatrix &matrix);
37
     /**
38
      ^{\star} Returns a vector containing the sum of the values of each lines.
39
      * @param matrix: the matrix containing the vectors to sum
40
41
      * @return Vector of all line sums
42
43
     IntVector lineSum(const IntMatrix &matrix);
44
45
46
     * Returns the vector of a matrix with the lowest sum of values.
      * If several vectors have the same sum, the function returns the one with the lowest index
47
      * @param matrix: the matrix to analyse
48
49
      * @return Line with smallest sum
50
      * /
51
     IntVector vectMinSum(const IntMatrix &matrix);
52
53
54
      ^{\star} Shuffles the vectors of a matrix without changing the vectors
      * @param matrix: the matrix to shuffle
55
56
57
     void shuffleMatrix(IntMatrix &matrix);
58
59
      ^{\star} Sorts a matrix (reverse order of the biggest line element)
60
      * @param matrix: the matrix to sort
61
62
63
     void sortMatrix(IntMatrix &matrix);
64
65
66
      * Computes the left to right diagonal sum and returns true if the matrix is valid (is square)
67
      * @param matrix: the matrix to sum the diagonal
      ^{\star} <code>@param</code> result: where left to right diagonal sum will be stored
68
69
      * @return True if the diagonal exists,
70
                 false if not
71
     bool diagLRSum(const IntMatrix &matrix, int &result);
```

library.h M. Vogel & T. Praz – HEIG-VD

```
73
74
75
     * Computes the right to left diagonal sum and returns true if the matrix is valid (is square)
     * @param matrix: the matrix to sum the diagonal
76
     * @param result: where right to left diagonal sum will be stored
77
78
     * @return True if the diagonal exists,
79
                false if not
80
81
    bool diagRLSum(const IntMatrix &matrix, int &result);
82
83
     #endif //LABO5_VECTEURS_LIBRARY_H
84
```

```
/*-----
 1
 2
    Filename
                   : main.cpp
 3
                   : Maëlle Vogel and Tobie Praz
    Authors
    Creation date : 01.12.2020
    Description
                   : This library provides functions to:
 6
                       - display a vector
 7
                      - display a matrix
 8
                      - check if a matrix is a square
9
                      - check if a matrix is regular
10
                      - return the size of the longest vector
11
                      - return a vector containing the sum of each vector
12
                      - return the vector with the smallest sum
13
                       - shuffle the vector order in the matrix
                       - sort the matrix by the biggest number in a vector
14
15
                       - sum the right to left diagonal /
16
                       - sum the left to right diagonal \
17
    Compiler
                   : Mingw-w64 g++ 8.1.0
18
19
    #include "library.h"
20
21
    #include <iostream>
22
    #include <string>
23
    #include <cctype>
24
    #include <algorithm>
2.5
    #include <numeric>
    #include <chrono>
26
27
    #include <random>
28
29
    using namespace std;
30
31
    //Utility functions
32
    string commaJoinInt(const string &a, int b) {
         return a + ", " + to_string(b);
33
34
35
36
    string commaJoinString(const string &a, const string &b) {
37
         return a + ", " + b;
38
39
40
    string vecToString(const IntVector &v) {
41
        string vec;
42
         if(!v.empty()) {
43
             //Join vector elements with ", "
44
             vec = accumulate(next(v.begin()), v.end(), to string(v[0]), commaJoinInt);
45
46
         return "(" + vec + ")";
47
     }
48
49
    int vecSize(const IntVector &v) {
50
         return v.size();
51
52
53
    int sum(const IntVector &v) {
54
        return accumulate(v.begin(), v.end(), 0);
55
56
57
    bool comparator(const IntVector &a, const IntVector &b) {
58
         return max element(a.begin(), a.end()) > max element(b.begin(), b.end());
59
60
61
     //Implementations
62
     ostream &operator<<(ostream &os, const IntVector &v) {</pre>
        cout << vecToString(v);</pre>
63
64
         return os;
65
    }
66
67
    ostream &operator<<(ostream &os, const IntMatrix &m) {
68
         vector<string> strings(m.size());
69
         //Convert matrix lines to string
70
         transform(m.begin(), m.end(), strings.begin(), vecToString);
71
72
         string mat;
```

```
73
          if(!strings.empty()) {
 74
              //Join matrix lines with ", "
 75
              mat = accumulate(next(strings.begin()), strings.end(), strings[0], commaJoinString);
 76
 77
 78
          cout << "[" << mat << "]";
 79
          return os;
 80
 81
 82
     bool isRegular(const IntMatrix &matrix) {
          int size = matrix.size();
 83
 84
          if(size) {
 85
              IntVector sizes(size);
 86
              //Get lines sizes
 87
              transform(matrix.begin(), matrix.end(), sizes.begin(), vecSize);
 88
              //Count the number of lines with same size as first line
 89
              return count(sizes.begin(), sizes.end(), sizes[0]) == size;
 90
 91
          return true;
 92
      }
 93
 94
      bool isSquare(const IntMatrix &matrix) {
 95
          return matrix.empty() || isRegular(matrix) && matrix[0].size() == matrix.size();
 96
 97
 98
      int maxCol(const IntMatrix &matrix) {
 99
          IntVector sizes(matrix.size());
100
          //Get lines sizes
101
          transform(matrix.begin(), matrix.end(), sizes.begin(), vecSize);
102
          //Fin biggest size
103
          return *max element(sizes.begin(), sizes.end());
104
105
106
     IntVector lineSum(const IntMatrix &matrix) {
107
          IntVector result(matrix.size());
108
          transform(matrix.begin(), matrix.end(), result.begin(), sum);
109
          return result;
110
      }
111
      IntVector vectMinSum(const IntMatrix &matrix) {
112
113
          IntVector sum = lineSum(matrix);
114
          //Get min line sum iterator and compute index
115
          int i = min element(sum.begin(), sum.end()) - sum.begin();
116
          return matrix[i];
117
118
119
      void shuffleMatrix(IntMatrix &matrix) {
120
          unsigned seed = chrono::system_clock::now().time_since_epoch().count();
121
          shuffle(matrix.begin(), matrix.end(), default random engine(seed));
122
      }
123
124
      void sortMatrix(IntMatrix &matrix) {
125
          sort(matrix.begin(), matrix.end(), comparator);
126
      }
127
128
     bool diagRLSum(const IntMatrix &matrix, int &result) {
129
          result = 0;
130
          if (isSquare(matrix)) {
131
              for (size_t i = 0; i < matrix.size(); ++i) {</pre>
132
                  result += matrix[i][matrix.size() - i - 1];
133
134
              return true;
135
136
          return false;
137
      }
138
139
      bool diagLRSum(const IntMatrix &matrix, int &result) {
140
          result = 0;
141
          if (isSquare(matrix)) {
              for (size_t i = 0; i < matrix.size(); ++i) {</pre>
142
143
                  result += matrix[i][i];
144
              }
```

library.cpp M. Vogel & T. Praz – HEIG-VD

```
145 return true;
146 }
147 return false;
148 }
149
150
```