CS-M48 Coursework

Francesca Madeddu
803623@swansea.ac.uk

November 19, 2014

Intro

All methods have been successfully tested. Here is the summary of all tests performed: look at the `Testing.cpp` class to see the actual implementation.

1 Test1

In the Test 1 constructors and get and set methods have been tested.

- **Constructor** tests:
  - create a empty matrix [1,1];
  - create a matrix MxN [1.2];
  - create a matrix by copying another one [1.3];
- **Get/Set mothods** tests:
  - populate a matrix with random values [1.4]
  - get a value of a matrix from a random position [1.5]

2 Test 2

In the Test 2 all operator overloaded methods have been tested. Because the `operator+` and `operator-` call respectively `operator+=` and `operator-=`, only the last two are present in the Testing.cpp class. Each method has been tested for failure case too.

- **Operator** tests:
  - sum of two matrix with different size (return first matrix value since the operation is not possible) [2.1]
  - sum of two matrix with same size [2.1]
  - subtraction of two matrix with different size (return first matrix value since the operation is not possible) [2.2]

  - subtraction of two matrix with same size [2.2]
  - multiplication of a matrix for a scalar [2.3]
  - multiplication of a square matrix for itself [2.4]
  - multiplication of a not square matrix for itself (return first matrix value since the operation is not possible) [2.5]
  - multiplication of a matrix MxN with a matrix NxM [2.5]
  - multiplication of a matrix MxN with M!=N for itselft (return first matrix value since the operation is not possible) [2.5]
  - is a matrix equal to itself? [2.6]
  - is a matrix equal to a different matrix? [2.6]

2.1 Note to implementation

All methods return a reference instead a pointer: in this way it is possible to concatenate operators. Operation like:

- `a = a + b;`
- `a = b + c + d;`

would not be possible because the return type of the operation would be a pointer, while the operator would expect a reference. Because it's not possible to return 0 to a reference, in case of failure the current matrix value (*this) not modified is returned instead.

3 Test 3

In the Test 3 all transformation methods have been tested.

- **Transformation methods** tests:
  - calculate the Transpose of a MxN matrix in a NxM matrix [3.1]
  - calculate the Transpose of a MxN matrix, with M!=N on itself (return first matrix value since the operation is not possible) [3.1]
  - calculate the Identity matrix of a square matrix [3.2]
  - calculate the Identity matrix of a not square matrix (return first matrix value since the operation is not possible) [3.2]

4 Other methods

In order to support the test class and the matrix implementation few more utility methods have been implemented:

- Utility methods:

- `void setNumRowsCols(const int, const int)`: set fields value of rows and cols for a given matrix;
- `bool isSameSize(const Matrix<T> &) const`: check if two matrix have the same size;
- `bool isSquare() const`: check if a matrix is square
- `void initRandom(const T) const`: inizialize a matrix with random value between 0 and a max value given as input parameter
- `void printContent() const` and `void printContent(const string) const`: both methods print the content of the matrix.