Assignment 2: Design and Implementation of Triangular Matrix Abstract Data Type

Dr. Shahadat Hossain
Department of Mathematics and Computer Science
University of Lethbridge

October 22, 2010

1 Introduction

In practical applications matrices that arise may have special "structures" and it usually pays (in storage needed and time to perform matrix operations) to take advantage of this information. Section 2 describes a structured matrix type, called "triangular matrix". Your assignment is to implement the triangular matrix abstract data type and test the correctness of your implementation. You may borrow ideas from the implementation of matrix ADT as described in the handout.

2 Triangular Matrix Abstract Data Type

A triangular matrix is a matrix where the nonzero elements are confined in *one half* of the matrix and the other half contains only zero. For this assignment we only consider matrices for which the number of columns is same as the number of rows. We can define two types of triangular matrices e.g., lower triangular and upper triangular as shown in Figure 1 where X

$$\begin{pmatrix} X & 0 & 0 & 0 \\ X & X & 0 & 0 \\ X & X & X & 0 \\ X & X & X & X \end{pmatrix} \quad \begin{pmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{pmatrix}$$

$$(a) \qquad\qquad (b)$$

Figure 1: (a) A lower triangular matrix. (b) An upper triangular matrix.

denotes an element which can be a zero or a nonzero. The matrix on the left is lower triangular and the matrix on the right is upper triangular. To implement a triangular matrix abstract data type we must utilize this special information i.e., we need not use memory to store matrix elements that are known to be zero. In other words, we need a dynamic array of size

$$\sum_{i=1}^{n} i = n(n+1)/2$$

to store only the elements that are marked X for an $n \times n$ triangular matrix. This way we save almost one half the space that is needed for a full matrix of dimension $n \times n$.

3 Assignment

In this assignment you are to design, implement, and test a class called *TriangularMatrix* using a one-dimensional dynamic array.

1. Explain (with the aid of an example) in your header file (as comments), how you have constructed the element mapping function. Here you must discuss whether the mapping is row-major or column-major. Also, your solution must be able to handle both lower and upper triangular matrices.

2. Implement the following member functions of the class and write the implementation in the file `TriangularMatrix.cc`.

   (a)
   ```
   explicit Matrix(int n=2);// create an n X n matrix
   Matrix(const Matrix& m); // copy constructor
   ~Matrix(); // destructor
   Matrix& operator=(const Matrix& m); // copy assignment
   ```

   (b)
   ```
   int size() const;      // return the size of the matrix
   Matrix& operator+=(const Matrix& m);
   Matrix& operator-=(const Matrix& m);
   Matrix& operator*=(const Matrix& m);
   Matrix& operator*=(double s) ; // scalar multiplication-assignment
   double operator()(int i, int j) const;          // r-value
   double& operator()(int i, int j);               // l-value
   Matrix operator+(const Matrix& lt, const Matrix& rt);
   Matrix operator-(const Matrix& lt, const Matrix& rt);
   Matrix operator*(const Matrix& lt, const Matrix& rt);
   Matrix operator*(const Matrix& lt, double s); // scalar multiplication
   Matrix operator*(double s, const Matrix& rt); // scalar multiplication
                                                 // (commutative)
   ostream& operator<<(ostream& out, const Matrix& x);
   void readMatrix(); //member function to read a matrix from users
   ```

3. Write a test program (in `testMatrix.cc`) showing the use of each member function that you have implemented. In your test program you should use `readMatrix()` (a member function) to input a matrix from the users interactively (i.e., number of rows, columns, whether lower or upper triangular, and the elements of the matrix). No file input/output is required.

Marks will be awarded as follows

1. Choice of appropriate data structures and explanation of the correct mapping function in the header file: 20 points.
2. Correct implementation of the member functions: total of 50 points.
3. Correct/appropriate pre- and post-conditions : 10 points
4. Use of appropriate comments and good coding style: 10 points.
5. The test program implemented according to the specification: 10 points.

Total: 100 points

4 Submission of Assignment

The submission of your assignment is to be done in two stages.

1. [**Due date: Monday October 25 by 11:59 pm. Weight:** 30%] In this part you submit the header file containing the constructors, the destructor, the copy assignment operator, the input function, and the output operator. Additionally, the implementation file containing the implementation of the functions specified in the header file and a test file with code to verify the functions implemented needs to be submitted.
2. [**Due date: Wednesday October 27 by 11:59 pm. Weight:** 70%] In this part you submit the complete implementation and testing code.