Scientific Programming

Assignment: Data Structure and STL.

Anand Kamble

amk23j@fsu.edu

17th November 2023

---

1. Introduction:

The SparseMatrix Operations program is designed to demonstrate the functionality of a sparse matrix class. The sparse matrix is represented by a vector of elements, each containing information about its row, column, and value. The program implements various operations such as scaling, multiplication, addition, and I/O operations for sparse matrices.

---

2. Implemented Operations:

2.1 Scaling Operation:

The program successfully demonstrates the scaling operation (scale * matrix). It initializes a SparseMatrix, scales it by a specified factor, and prints the result.

2.2 Matrix Multiplication Operation:

Matrix multiplication (matrix * matrix) is implemented and tested with two example matrices. The result is a new SparseMatrix with the product of the input matrices.

2.3 Matrix Addition Operation:

The program supports matrix addition (matrix + matrix). It creates two example matrices, adds them, and outputs the resulting SparseMatrix.

2.4 I/O Operations:

The program showcases the ability to read from and write to files using the << and >> operators. It saves a SparseMatrix to a file and reads it back, demonstrating successful I/O operations.

3. Random SparseMatrix Generation:

A function is provided to generate a SparseMatrix with random values in the range [0, 1]. The random number generator is initialized, and random values are assigned to the matrix elements.

---

4. Testing and Results:

The program's main function thoroughly tests the implemented operations. It creates matrices, performs operations, and outputs the results. Additionally, the I/O operations are tested by saving a matrix to a file, reading it back, and displaying the results.

5. Conclusion:

The SparseMatrix Operations program successfully implements and tests various matrix operations, including scaling, multiplication, addition, and I/O operations. The provided random number generator enhances the program's flexibility for generating matrices with random values.

6. Recommendations for Future Improvements:

Implement error handling for invalid user inputs.

Explore optimization opportunities for matrix multiplication.

Enhance the program's user interface for a more interactive experience.