D1166570 Levi

Programming Assignment 6 Report:

Linear Equation System Solver

Introduction:

Embarking on the journey of implementing a Linear Equation System Solver in C++ provided an enriching experience into the realms of object-oriented programming and matrix algebra. This report outlines the steps undertaken, challenges faced, and insights gained throughout this assignment, which required defining a `Matrix` class and its derived classes `Vector` and `SMatrix` to solve systems of linear equations.

Dynamic Memory Management:

A core aspect of this assignment involved efficient memory management. In C++, dynamic memory allocation is crucial for handling objects that are created and destroyed during runtime. The `Matrix` class, though simple in structure, required careful consideration of memory handling, particularly when dealing with matrix operations during the solution of linear equations.

Implementing Matrix and Vector Operations:

The primary task was to implement fundamental matrix and vector operations. This included:

- Addition, subtraction, and multiplication of matrices.
- Handling vectors as special cases of matrices.
- Randomly generating matrices and vectors with values between 0 and 1.

Dealing with Edge Cases:

Handling edge cases such as division by zero and invalid matrix dimensions was critical. Implementing checks within the overloaded operators ensured the robustness of the classes, preventing erroneous operations and maintaining integrity under various conditions.

Solving Linear Equation Systems:

To solve the linear equation system $AX = C$, where A is a square matrix and X and C are vectors, the application program was designed to:

1. Input n as the rank of the system, ensuring $0 < n \leq 10$.
2. Randomly generate the coefficient matrix A and constant vector C.
3. Print the generated matrix A and vector C.
4. Solve the linear equation system to obtain the solution vector X.
5. Verify the solution by computing $AX - C$ and checking whether it is the zero vector within an error tolerance of six decimal places.

Testing the Implementation:

To validate the functionality of the classes and the solver program, comprehensive testing was conducted. This included:

- Performing a series of matrix and vector operations.
- Generating random matrices and vectors to simulate different systems of equations.
- Ensuring precision up to six decimal places as specified in the assignment.

Organizing the Code:

Adopting a modular approach, the code was organized into separate files:

- `matrix.h` and `matrix.cpp` for the `Matrix` class definition and

implementation.

- `vector.h` and `vector.cpp` for the `Vector` class definition and implementation.
- `smatrix.h` and `smatrix.cpp` for the `SMatrix` class definition and implementation.
- `linear_equation_system_solver.cpp` for the application program to solve linear equations.

This structure enhanced maintainability and readability, facilitating easier debugging and future enhancements.

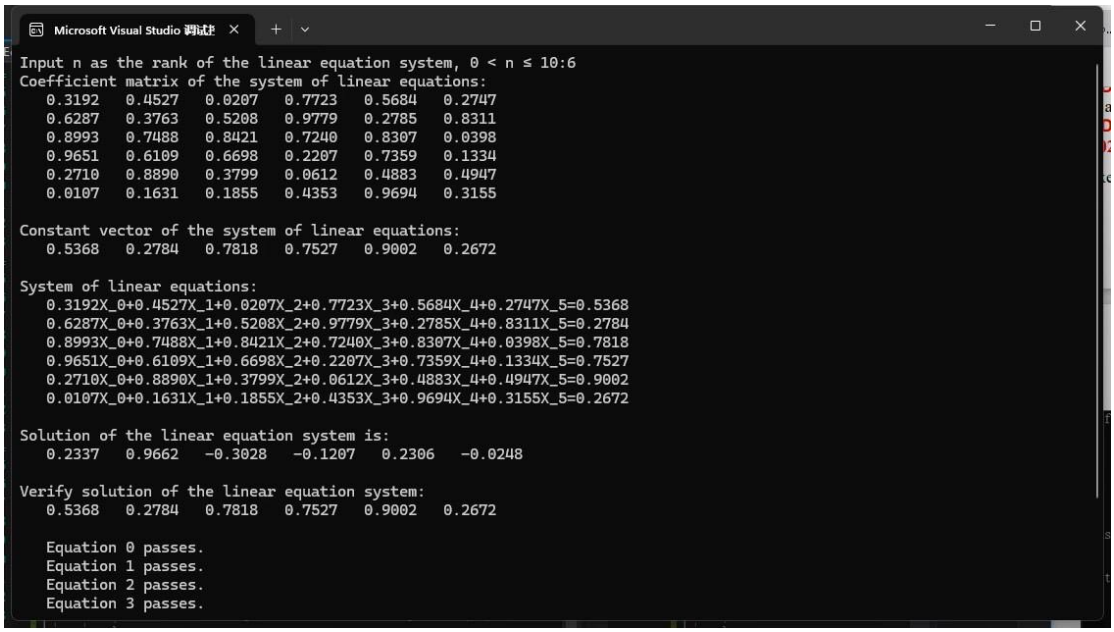
Reflections and Conclusions:

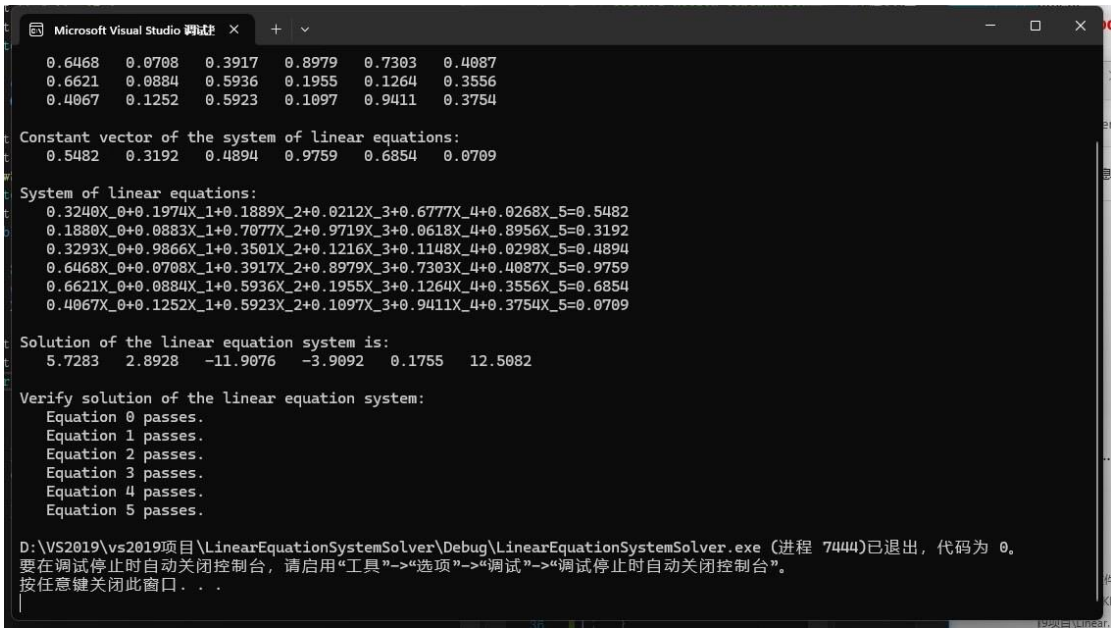
This project was a profound exercise in understanding the intricacies of matrix operations and object-oriented programming in C++. It highlighted the power of C++ in managing complex data structures and performing sophisticated operations with ease. Each step of the implementation deepened my appreciation for the language and refined my programming skills.

In conclusion, this assignment was not just an academic requirement but a significant learning experience. It provided valuable insights into advanced programming concepts and prepared me for more challenging endeavors in the future. As I move forward, I carry with me the lessons learned and a renewed curiosity to explore further into the realms of computer science.

The following includes the result of my assignment.

D1166570 Levi





2024/5/29