Reflection on COBOL Re-engineering Process

The process of modernizing legacy COBOL code revealed several important lessons about software engineering and maintenance. Updating statmold.cob to statmeasure.cob required careful consideration of both technical and design aspects.

One of the primary challenges was updating the I/O handling from fixed filenames and sentinel-based termination to dynamic filenames and proper end-of-file detection. The original approach relied on a specific sentinel value (999999.99) to terminate input processing, which created a limitation on valid data values. Modern COBOL provides better constructs like AT END clauses that make the code more robust and flexible.

Another significant challenge was restructuring the calculations to ensure proper handling of edge cases. When implementing geometric and harmonic means, I had to consider mathematical constraints - geometric means require positive values, and harmonic means cannot handle zeros. Adding proper error checking and reporting was essential to prevent runtime errors.

Converting the COBOL programs to free format required careful attention to syntax details. The key modifications included:
1. Adding proper period terminators to all statements
2. Converting continuation lines to single-line statements
3. Ensuring proper statement termination within conditionals
4. Adapting to the more flexible columnar format of free COBOL

The conversion to free format improved code readability significantly while maintaining the same functionality. The successful compilation and execution of both programs with the "-free" flag confirms that the conversion process preserved all the original features.

The most important lessons learned were:
1. Well-structured code with meaningful names and sections dramatically improves readability and maintainability
2. Modern COBOL features like END-IF, END-READ, and COMPUTE make code clearer and less error-prone
3. Proper error handling is crucial, especially when dealing with mathematical operations
4. Thorough testing with various input datasets reveals edge cases that might otherwise be missed
5. Free format COBOL provides significant improvements in readability and maintainability

For further enhancement, the program could benefit from additional improvements:
1. Adding options for output format customization
2. Implementing more statistical functions like median or quartiles
3. Creating an option to process multiple files in a single run
4. Adding data validation to handle improperly formatted input

Overall, this re-engineering exercise demonstrates the importance of modernizing legacy code rather than simply replacing it. The COBOL language, when properly used with modern constructs and free format, can produce maintainable and efficient programs that meet current requirements while preserving the original business logic.