In this project, we designed a C++ program to handle complex number arithmetic using a class called Complex. The class implements arithmetic operations such as addition, subtraction, multiplication, and division of complex numbers. Additionally, it supports methods to retrieve and set the real and imaginary parts of complex numbers, as well as overloaded operators for assignment and comparison.

Precision Loss in Arithmetic Operations: Issue: Due to floating-point arithmetic, there may be precision loss in complex number operations, especially during multiplication and division. Solution: Implement error-handling mechanisms such as rounding or using higher precision data types to mitigate precision loss. Additionally, consider using libraries like Boost for arbitrary-precision arithmetic.

Input Validation for Division: Issue: Division by zero is not handled explicitly, which can lead to runtime errors or undefined behavior. Solution: Add input validation to check for zero in the denominator before performing division operations. Throw an exception or return an error message to indicate invalid inputs.

Handling Complex Roots in Quadratic Equations: Issue: When solving quadratic equations with complex roots, comparing floating-point values for verification may lead to inaccuracies due to limited precision. Solution: Implement a tolerance mechanism to account for small variations in results. Compare the absolute difference between the calculated value and zero to determine if it is within an acceptable range, such as 0.000001.

Conclusion:

By addressing potential issues such as precision loss, input validation, and handling complex roots, we ensure the reliability and accuracy of our complex number arithmetic implementation in C++. These solutions enhance the robustness of the program, making it suitable for various mathematical applications.