Report for assignment 5 D1262075陳映聿

**Complex Number Handling: C vs. C++**

In C, managing complex numbers necessitates structures and functions due to the lack of operator overloading or classes. Structures are used to store real and imaginary parts, with functions implementing operations like addition, subtraction, multiplication, and division.

Conversely, C++ simplifies complex number manipulation through classes and operator overloading. Classes encapsulate real and imaginary components, while operator overloading allows direct use of operators for arithmetic operations. This leads to more intuitive syntax. Additionally, C++ supports encapsulation, enhancing data protection and organization.

**Key differences:**

C uses structures for complex numbers, while C++ uses classes.

C requires functions for operations, while C++ uses operator overloading.

C++ supports encapsulation for better data management.

**Pros and Cons of C++ Programming**

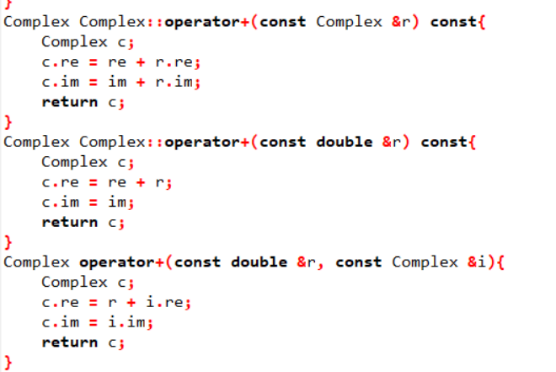
C++ offers numerous advantages. It supports object-oriented programming, promoting data abstraction, encapsulation, inheritance, and polymorphism for code reusability and modularity. Operator overloading enhances readability, especially for custom data types like complex numbers. The STL provides a rich set of template classes and functions for common tasks, reducing the need for custom code. Robust error handling through exceptions and memory management options add to its appeal.

However, C++ complexity poses challenges for beginners. Features like exception handling and virtual functions may impact performance. Object-oriented programming can lead to verbose code, particularly for simple tasks. Mixing C and C++ can result in complex codebases.

**Complex\_overloading.h:**

in this code define a class name Complex, and there are 8 friend functions including input and output operations and arithmetic operations. In the public part, there are two constructors, one with default argument (set value to 0) and another copy constructor. And some arithmetic operators for addition, subtraction, multiplication, and division, both with complex numbers and doubles. Lastly in the private define real part and imaginary part of complex numbers.

**Complex\_overloading.cpp:**

In this code, implement different arithmetic functions (complex, complex double, double complex). Write the use of every functions. Take functions of addition for example: 

**Quadratic\_equation\_verifier.cpp:**

In this code, first make a print equation function to print out the equation with coefficients. Then in the main function, use the function defined to calculate the result using discriminant to consider if there is imaginary part of the roots or not, at the same time use setComplex to form it and print out. Lastly, do the verification, consider the precision error up to six digits after decimal point.