COMPLEX ARITHMETIC IN THE STANDARD C++ LIBRARY

● Calling Procedure
To activate complex number support in the standard library, add the header:

```cpp
#include<complex>
```

(The mathematical functions defined in the `<cmath>` library become accessible, too.)

● Declaring Complex Variables
The `complex` library defines *templates* for complex numbers in the forms:

```cpp
complex<float> Variable;
complex<double> Variable;
complex<long double> Variable;
```

This declares a complex variable of single precision (`float`), double precision, or extra precision (`long double`). What that actually means in practice depends on the computer system you use. A good bet is to use the `complex<double>` form.

● Assigning Complex Variables
To assign a value to a complex variable (here for `double`), use one of these statements:

```cpp
Variable = complex<double>(double RealPart, double ImagPart);
Variable = double RealPart;
```

`RealPart` and `ImagPart` can both be either floating-point constants (like `1.0` or `-2.5` etc.), or variables of type `double`. In the second form, the imaginary part of the complex variable is set to zero. You can declare and assign in one combined step:

```cpp
complex<double> ImagUnit = complex<double>(0.0, 1.0);
```

● Operators and Complex Numbers
Basic arithmetic operations using complex numbers are performed in pretty much the way that you would expect them. Defined operators are:

- Assignment: `=`
- Arithmetic: `+ - * /`
- Combined: `+= -= *= /=`
- Comparison: `== !=`
- Input/Output: `<< >>`

The arguments of these operators must be either of type `complex<double>` or `double`. E.g.,

```cpp
complex<double> z1 = complex<double>(-1.0, 2.0);
double x = 1.0;
complex<double> Result = z1 + ImagUnit;
Result += x;
cout << Result;
```

should yield the output:

```
(0.0,3.0)
```
• **Mathematical Functions by Type**

  ✗ **Cartesian and Polar Forms:**

  - `double real(complex<double> z)`
    - Extract real part $Re(z)$ of complex number.
  - `double imag(complex<double> z)`
    - Extract imaginary part $Im(z)$ of complex number.
  - `double abs(complex<double> z)`
    - Determine modulus $|z|$ of complex number.
  - `double arg(complex<double> z)`
    - Determine argument (phase angle) $\arg z$.
  - `complex<double> polar(double r, double phi)`
    - Define complex number in polar form, $z = r \cdot e^{i \phi}$.

  ✗ **Absolute Values and Complex Conjugate:**

  - `double norm(complex<double> z)`
    - Determine square $|z|^2 = zz^*$ of complex number.
  - `complex<double> conj(complex<double> z)`
    - Form conjugate complex number $z^*$.

  ✗ **Root, Power, Exponential, and Logarithmic Functions:**

  - `complex<double> sqrt(complex<double> z)`
    - Calculate the square root of $z$.
  - `complex<double> pow(double x, complex<double> z)`
  - `complex<double> pow(complex<double> z, int n)`
  - `complex<double> pow(complex<double> z, double x)`
  - `complex<double> pow(complex<double> z, complex<double> w)`
    - Calculate powers $x^z, z^n, z^x, z^w$ (various combinations).
  - `complex<double> exp(complex<double> z)`
    - Calculate exponential function $e^z$.
  - `complex<double> log(complex<double> z)`
    - Calculate natural logarithm $\ln z$.

  ✗ **Trigonometric Functions:**

  - `complex<double> sin(complex<double> z)`
    - Calculate sine of argument $\sin z$.
  - `complex<double> cos(complex<double> z)`
    - Calculate cosine of argument $\cos z$.
  - `complex<double> tan(complex<double> z)`
    - Calculate tangent of argument $\tan z$.

  ✗ **Hyperbolic Functions:**

  - `complex<double> sinh(complex<double> z)`
    - Calculate hyperbolic sine of argument $\sinh z$.
  - `complex<double> cosh(complex<double> z)`
    - Calculate hyperbolic cosine of argument $\cosh z$.
  - `complex<double> tanh(complex<double> z)`
    - Calculate hyperbolic tangent of argument $\tanh z$. 