# cpx — C++ class for complex numbers

v2.1(β) / 07.03.2022

Mihai-Tiberiu Dima, Maria Dima
Meshcheryakov Lab. Information Tek, JINR, Russia

**Header**: `#include "cpx.hh"`

**Libraries**: `libcpx.a` (static) linux-2.6.32-504.12.2.el6.x86_64
`libcpx.so` (dynamic) gcc-4.4.7 20120313

**Variables** (public):

- `x = scalar` ..... cartesian, real part (polymorphic)
- `y = scalar` ..... cartesian, imaginary part (polymorphic)

**Constructors** (public)

- `cpx<double> a` ..... initialisation to zero
- `cpx<int> a(1,0)` ..... initialisation to given value
- `auto a = b` ..... initialisation to b (cpx, or scalar)
- `auto a(b)` ..... initialisation to b (cpx, or scalar)

**Assign** (public)

- `cpx<int> a; a = b` ..... initialisation to b (cpx, or scalar)

**Cast** operators

- `int(z)` ..... conversion cpx<scalar> → int
- `double(z)` ..... conversion cpx<scalar> → double
- `long double(z)` ..... conversion cpx<scalar> → long double

**Negative** (friend)

- `auto a = -b` ..... initialisation to -b (const&, or &&)

**Conjugate** (friend)

- `auto a = ~b` ..... initialisation to b* (const&, or &&)

**Algebraic operators** (friends)

- `auto a = b+c` ..... b, c are (const&, or &&), (cpx, or scalar)
- `auto a = b-c` ..... b, c are (const&, or &&), (cpx, or scalar)
- `auto a = b*c` ..... b, c are (const&, or &&), (cpx, or scalar)
- `auto a = b/c` ..... b, c are (const&, or &&), (cpx, or scalar)
- `auto a = b^c` ..... b, c are (const&, or &&), (cpx, or scalar)
  exponentiation

- `auto a += b` ..... b is (const&, or &&), (cpx, or scalar)
- `auto a -= b` ..... b is (const&, or &&), (cpx, or scalar)
- `auto a *= b` ..... b is (const&, or &&), (cpx, or scalar)
- `auto a /= b` ..... b is (const&, or &&), (cpx, or scalar)

**Functions** (friend)

- `fabs(z)` ..... complex norm
- `phi(z)` ..... complex phase [ 0 … 2π]
- `phi2(z)` ..... complex phase [-π … +π]
- `sqrt(z)` ..... square root, + solution
- `cbrt(z)` ..... cubic root, 1st solution
- `log(z)` ..... log, minimal phase convention
- `exp(z)` ..... real part and Euler for imaginary part

**Print** (friend)

- `cout << z << endl;` ..... print z = a + ib
- `cout << boolalpha << z << endl;` ..... print scalar type appended

**Usage** examples

- `return cpx<double>(r,c)` ..... return temporary from function call
- `auto z = e^(cpx<int>(0,1)*pi)` ..... $z = e^{i\pi}$

**Note**

- protect operator ^ with ( … ) due to its low precedence
  `cout << e^(j*pi) << endl;` needs to be `cout << ( e^(j*pi) ) ...`

## Description

The ***cpx*** class is a very slim (2 variables, constructors, cast operators) templated C++ class. The huge number of non-class operators (27+1205) are `friend`, saving an extra variable (`this`) in the call, for somewhat higher runtime expediency. A deeper reason is due to templated coding, each operator function needing ca. 7 implementations, in order to accomodate *quasi-polymorphism*.

Quasi-polymorphism means the package mimics polymorphism for the usual scalar types used in science and engineering. Statements such as:

```
auto z = double(1) + cpx<int>(3,0) ;
```

benefit of the templated function type-calculator to determine the output type as `cpx<double>`.

The class overloads `fabs` to calculate the norm – and has 2 functions, `phi(z)` and `phi2(z)` the first in mathematic mode [0 … 2π] and the second, engineering-wise.

The class comes with all instantiation combinations for `int, float, double, long double`.

The ***makefile*** is banale, however with full pfledged functionality: `make libs`, `make test`, `make run`, `make clean`.

The class comes with 4 examples and 1 application example.