Proposal for C2Y
WG14 N3452

Title:	Complex literals warning
Author, affiliation:	C FP group
Date:	2024-12-04
Proposal category:	Editorial
Reference:	N3215, N3301, N3390

Previous email and documents, e.g. N3215, have pointed out problems involving signed zeros and infinities when imaginary types are not available. The optional imaginary types in C99 through C23 are generally not implemented and have been removed from draft C2Y. Complex literals (6.4.5.3) add an implicit 0 real part to pair with the explicit imaginary part to compose the complex value, and it is this 0 that confounds signed zeros and infinities. The macro `I` (which N3390 proposes to remove) when defined to be `_Complex_I` has the same problems. The suggested change below adds a note to warn users about the problems and shows a means to avoid them.

Suggested change:

To 6.4.5.3 #12, append a second note (and number the first note):

> NOTE 2 Since complex literals are of complex type and have a (positive) zero real part, their use is problematic where signed zeros and infinities matter. For example, with ISO/IEC 60559 arithmetic and its default rounding,
>
> | | | |
> |---|---|---|
> | `-0.0 + 1.0i` | yields | $-0 + (+0 + i) = +0 + i$ |
> | `-1.0i` | yields | $-(+0 + i) = -0 - i$ |
> | `INFINITY * 1.0i` | yields | $\infty \times (+0 + i) = \text{NaN} + \infty i$, and raises the invalid floating-point exception when evaluated at execution time |
>
> The `CMPLX` macros in `<complex.h>` can be used to straightforwardly obtain particular values involving signed zeros and infinities. Corresponding to the cases above,
>
> | | | |
> |---|---|---|
> | `CMPLX(-0.0, 1.0)` | yields | $-0 + i$ |
> | `CMPLX(0.0, -1.0)` | yields | $+0 - i$ |
> | `CMPLX(0.0, INFINITY)` | yields | $+0 + \infty i$ |