Pre-Prague 2020 Paper Review

- C++ Numerics Work In Progress: http://wg21.link/P1889R1
  - Status: in Study Group
  - Goal: Numerics TS or WD, see http://wg21.link/P2004r0
  - Notes:
    - Functions that detect or allow control over:
      - Overflow
      - Rounding
      - [Lossy] conversion
      - Clamp
    - ^ see http://wg21.link/p0103 and http://wg21.link/P0105
    - wide_int (see below)
    - Rational arithmetic see http://wg21.link/n3611
    - Parametric aliases (ie. std::least_2uint<14> -- at least 14bits of unsigned in)
    - Unbounded integer see http://wg21.link/N4038
- Proposal to add wide_int template class: http://wg21.link/P0539R5
  - Status: in Study Group
  - Goal: Numerics TS or WD, see http://wg21.link/P2004r0
  - Notes:
    - >64bit fixed width un/signed integers and all operations
    - Make it appear as if std::[u]int{128, 256, 512}_t exists.
- C++ Approach to Physical Units: http://wg21.link/P1935R2
  - Status: in Study Group / Library Incubator
  - Goal: WD
  - Notes:
    - Explicitly for critical reliability
    - Stick to physical units
    - Extensible beyond
    - Large survey of previous works and trade-offs
      - Hard to debug errors
      - Unnecessary conversions to base unit (ie. m or s)
      - Can't represent large ratios (ie. astrophysics)
      - Extensibility to new systems or dimensions
      - Verbose code
    - Leverage C++20 concepts and inheritance-based specialization
      - Give shorter code (concepts terse syntax)
      - Give understandable error messages
      - Not require unnecessary/runtime conversions to base unit
      - Extensible to new systems with different ratios
      - New dimensions as concepts
    - New ratio class accepts exponent parameter
    - Prefer constants over user defined literals
    - Relative v. absolute is still open (ie. C v K)

            - Relative 0C + Relative 0C = Relative 0C
            - Absolute 0C + Relative 0C = Absolute 0C
            - Absolute 0C + Absolute 0C = 273.14C ??
        - May not be able to use std::chrono::duration as-is
- Proposal to add linear algebra support: http://wg21.link/P1385R5
    - Status: Library Incubator
    - Goal: C++23
    - Notes:
        - **Not** for scientific computing/supercomputers
        - Performance comparable to Eigen
        - API mimics mathematical notation
        - Configurable data ownership, lifetime, layout, access
        - Optimizable
        - Concepts:
            - Engines: allocator-like type that determines ownership, lifetime
            - Element promotion traits
            - Engine promotion traits
            - Arithmetic traits: does the actual computation
            - Operation traits: combines engine, promotion, arithmetic traits
            - Operation selector traits: selects operation traits
        - See examples
        - Still integrating http://wg21.link/p1673 (BLAS) as backend
- Programming Language Vulnerabilities for Safety Critical C++: http://wg21.link/P1706R2
    - Status: WG21 + WG23
    - Goal: MISRA
    - Notes:
        - Not specifying subsets, but giving guidance to WG23 and MISRA
        - Examine MISRA C++ Draft
        - MISRA will be more up to date on recent standards
        - WG23 C++ language vulnerabilities see http://www.open-std.org/JTC1/SC22/WG23/docs/ISO-IECJTC1-SC22-WG23_N0908-tr24772-10-C++-after-mtg-66-20191107.docx