std::atomic<>

- Ensure “tear free” & synchronised manipulation of shared data
- May use traditional locks if data-type cannot be manipulated atomically in hardware. Always check **`std::atomic<>::is_always_lock_free`**!
- Only a subset of manipulations are supported:
  - Store
  - Load
  - Atomic addition/subtraction
  - exchange/compare-exchange
- More info here: https://herbsutter.com/2013/02/11/atomic-weapons-the-c-memory-model-and-modern-hardware/

atomic Summary

- Scenario:
  - Multiple threads may need to mutate the data
- Trade-off:
  - Data is small: `std::atomic<>::is_always_lock_free == true`
  - Only certain operations are allowed
- Examples:
  - Sharing small data between threads
  - Gain values, level meters, automation values, parameters etc.