False Sharing

This lesson gives an overview of a false sharing problem which might occur during the implementation of concurrency in C++.

When a processor reads a variable such as an `int` from main memory, it will read more than the size of an `int` from memory; the processor will read an entire cache line (typically 64 bytes) from memory. False sharing occurs if two threads read different `int`'s at the same time, `a` and `b` that are located on the same cache line. Although `a` and `b` are logically separated, they are physically connected. An expensive hardware synchronization on the cache line is necessary because `a` and `b` share the same one. The result is that you will get the right results, but the performance of your concurrent application decreases.

`std::hardware_destructive_interference_size` and `std::hardware_constructive_interference_size` with C++17

Both functions let you deal in a portable way with the cache line size. `std::hardware_destructive_interference_size` returns the minimum offset between two objects to avoid false sharing and `std::hardware_constructive_interference_size` returns the maximum size of contiguous memory to promote true sharing.