Atomic Types Solutions

Atomics

- Write a task function that uses a for loop to increment a global variable 100,000
- Write a program that runs this task in concurrent threads and displays out the final value of the variable
- Increase the number of threads until you see signs of data corruption
- Make the variable atomic
  - Do you still get data corruption?
- Make the variable volatile
  - Do you still get data corruption?
- Briefly explain your results

Atomic Types and Operations Solutions

- Non-atomic version
  - There is a data race. Threads interfere with each other when incrementing the counter. The result is incorrect
- Atomic version
  - Fetching the counter, incrementing it and storing the new value is done as a single uninterruptible operation. There is no interference between threads and the result is correct
- Volatile version
  - The volatile keyword has no significance in multi-threaded C++ programs. This is equivalent to the non-atomic version