System Details

MacOS, Xcode compiler, C++20

Program Overview

In short, the program creates a buffer in which a producer and a consumer assign and withdraw items from the indices of the buffer in parallel fashion, or using multi-threading.

Expanding on the above, the program creates 2 threads, the producer, and the consumer.
The producer thread generates a random number and assigns it to an index of the buffer. However, if the next index of the buffer is the same index as the consumer, the producer will wait until the consumer's index moves on (Due to parallelism, we don't want the producer and the consumer to access the same index at the same time). Nonetheless, if the next index is unoccupied, the producer will move to the next index and assign another random number. It is worth mentioning we use modulus operator % to circle back to the first index when we reach the last index.
The consumer, similarly, will not execute if the producer's current index is the same as the consumer's index. But once the current index is unoccupied, the consumer will retrieve the index's value, and re-initialize it to 0. Thereafter, the consumer will move to the next index.
The producer-consumer multithreading operation will continue until the counter limit is reached.

Analyzing the output, we can see that indices of the buffer are printed, in this case of size 5. Then, the producer assigns numbers, at the same time, the consumer removes numbers. At some point, the output becomes distorted, this is because sometimes the producer and consumer are overlapping each other, however, in the future, this can be fixed by implementing some type of threading synchronization.