**Name**- Smriti Islur

# Multithreaded Application

**Design Choices**

The program is designed to efficiently process data concurrently from two files using multithreading. Key design choices include the use of mutexes to synchronize access to shared data and the implementation of a thread function (processDataThread) to handle data processing.

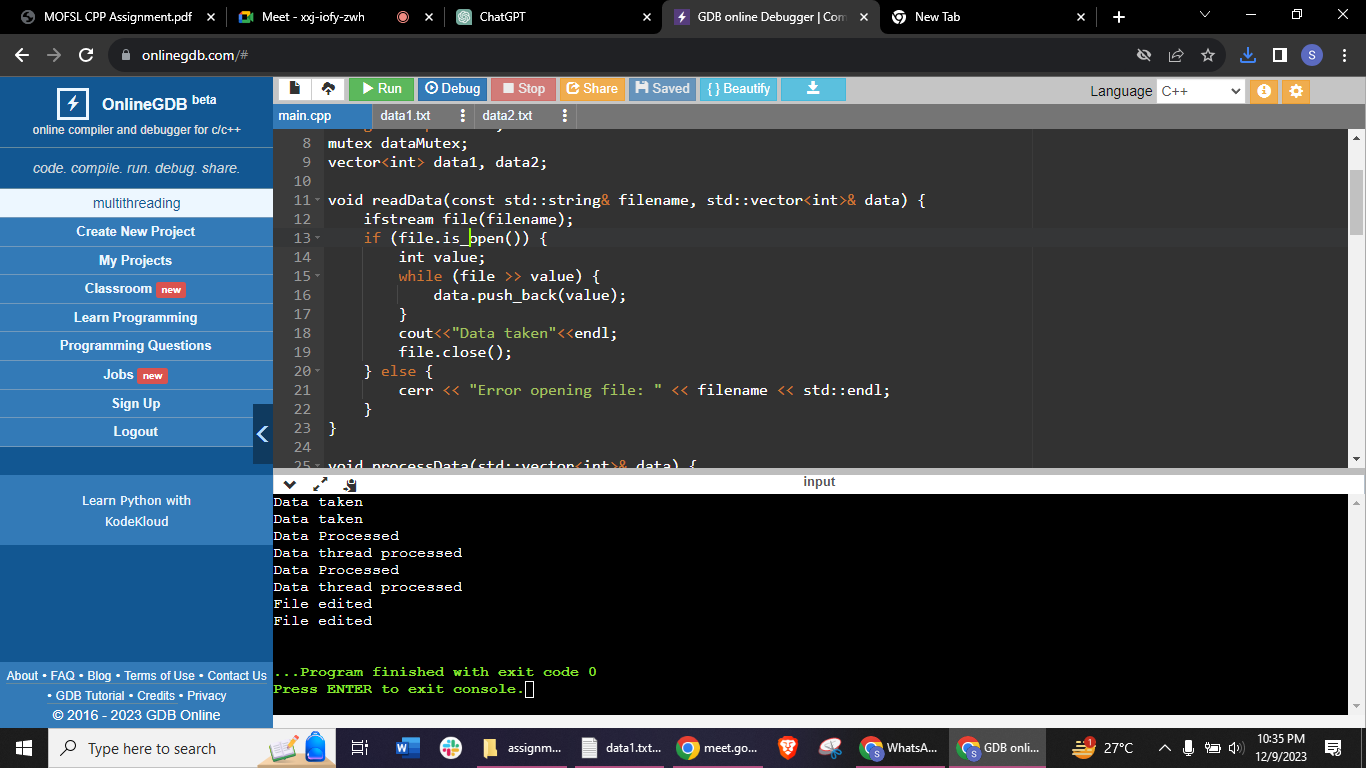
**Synchronization Mechanisms**

Mutexes (dataMutex) are employed to ensure exclusive access to shared data structures (data1 and data2). This prevents race conditions and ensures data integrity during concurrent operations. The use of lock\_guard simplifies the locking and unlocking process, reducing the likelihood of errors.

**Performance Considerations**

Efficient multithreading is achieved by dividing the workload across multiple threads. The lock\_guard and mutexes are employed to minimize the critical section and avoid contention. The design focuses on simplicity and readability while maintaining thread safety.

**Output**



**Conclusion**

The program successfully demonstrates multithreading, data processing, and file I/O synchronization in C++. By using mutexes and appropriate synchronization mechanisms, the design ensures data integrity and efficient concurrent processing.