The code demonstrates the implementation of a queue data structure using a doubly linked list. Here's a breakdown of the techniques used in each file:

1. Node.h:

Defines the Node class, which represents a node in the doubly linked list.

Includes private data members elem (to store the node's data), prev (to point to the previous node), and next (to point to the next node).

Declares two constructors: a default constructor and a constructor with a data element parameter.

2.Node.cpp:

Implements the constructors of the Node class.

Initializes the data members of the Node class in the constructors.

3.IQueue.h:

Declares the IQueue class, representing the integer queue.

Includes private data members head and tail, which are pointers to the head and tail of the queue, respectively.

Declares member functions for various queue operations: enqueue, dequeue, headElem, isEmpty, getSize, getHead, getTail, and printHeadToTail.

Includes a friend declaration to allow the IQueue class to access private members of the Node class.

4.IQueue.cpp:

Implements the member functions of the IQueue class declared in the header

file.

Defines the behavior of queue operations such as enqueue, dequeue, isEmpty, getSize, and others.

Utilizes the Node class to create and manipulate nodes in the doubly linked list.

5.IQueueMain.cpp:

Implements the main() function, which serves as the entry point of the program.

Creates an instance of the IQueue class and performs enqueue and dequeue operations on it.

Generates random values for the number of trials, enqueue operations, and dequeue operations to demonstrate the functionality of the queue.

Overall, the code demonstrates encapsulation and modularity by using separate header and implementation files for the Node and IQueue classes. It also illustrates the usage of doubly linked lists to implement a queue data structure.