The code is composed of three main components:

CMessageQueue: This class represents a message queue that can be used to send and receive messages. It uses a WriterReaderAutomata to handle the asynchronous reading and writing of messages.

CQueueBroker: This class represents a message queue broker that manages a pool of CMessageQueue objects. It can be used to bind and unbind ports, connect to other brokers, and accept new connections.

WriterReaderAutomata: This class is a state machine that manages the asynchronous reading and writing of messages. It transitions between states depending on the current state of the message queue.

The code uses an event-oriented design to handle the asynchronous reading and writing of messages. The WriterReaderAutomata class is responsible for transitioning between states and handling the I/O operations. The CMessageQueue and CQueueBroker classes provide a higher-level API for sending and receiving messages.

CMessageQueue:
CMessageQueue represents an event-driven message queue built on a channel. It enables asynchronous communication between different components in a distributed system.

Connecting:

A CMessageQueue is established when a connect matches an accept, creating a fully connected state.
When connecting, the provided name is that of the remote broker, and the port is from the accept on the remote broker.
Connect and accept operate in a symmetrical rendezvous manner.
Two cases are distinguished during connecting: (i) no accept yet, or (ii) no such broker.

The send method is used to write a byte array to the channel and so calls the write method on the WriterReaderAutomata object.
The channel is treated as a stream, so the write operation is performed one byte at a time.

Reading:

The setListener method sets a listener for received messages on the queue and calls the read method on the WriterReaderAutomata object.

Closing:

The close method is used to close the channel associated with the message queue.
CQueueBroker:
CQueueBroker extends the QueueBroker and acts as an interface between the CMessageQueue instances and a lower-level broker. It facilitates asynchronous communication over a fully connected network.

Connecting:

CQueueBroker manages the connection between brokers, and it uses an Executor for asynchronous operations.
The bind method establishes a connection by accepting on a specified port and notifying the AcceptListener.
The unbind method disconnects from a specified port.
Disconnecting:

A channel can be disconnected asynchronously, with the disconnect method handling disconnection from either side.
If the remote side disconnects, there might be bytes in transit, and the local side remains connected until those bytes are read.

WriterReaderAutomata:
WriterReaderAutomata manages the asynchronous reading and writing of bytes on a channel associated with a CMessageQueue.

Reading and Writing:

Uses a state machine with READ_SIZE, READ_CONTENT, WRITE_SIZE, and WRITE_CONTENT states.
Reading and writing are performed asynchronously, and the machine handles partial operations.
Disconnecting:

Handles the asynchronous nature of channel disconnection, ensuring proper handling of bytes in transit.