IRC Chat Program (13March2019)

STRUCTURE:

User Command

Handler

noSQL (Authenticate)

User Database

Incoming Client Connection

Local

CMD

Handler

Main

User Session

(New thread)

TCP Listener

(Thread)

Chat (Atomic)

Connect/Disconnect

Input

Chat Sync

TCP Requester

Main

Sender/Receiver

(Thread)

Local Command

Handler

BITFIELD:

* Reserve 8 bits for server handling. First 2 (00) will be (00) for authenticate, (01) for authenticated success, (10) for updates, (11) for requests. Rest of the bits are reserved for future use.
* todo

DOCUMENTATION:

Debugger:

Debug::WriteCriticalMessage(const string message)

This writes message at **critical** level.

Debug::WriteImportantMessage(const string message)

This writes message at **important** level.

Debug::WriteInformationalMessage(const string message)

This writes message at **informational** level.

Debug::SetDebugLevel(DebugLevel level)

This sets the level for the program with enum class DebugLevel with **NoLevel**, **CriticalLevel**, **ImportantLevel**, and **InformationalLevel**.

Debug::m\_ConsoleLog

Vector pointer to all messages outputted by the debugger.

Debug::OutLogDump()

Creates a text file IRC\_Program\_Log.txt with m\_ConsoleLog.

IRC\_Program:

int IRC\_Program(const std::vector<std::string>& args)

Starts the normal routine. Args from main

NetworkFactory:

void StartListening(const int& tickRate, std::atomic<bool>& endProgramIndicator, std::atomic<bool>& endProgramIndicatorResult)

Used in a thread for making the program listen for TCP connections. Spawn another thread per sessions.

DEPENDENCIES:

For building in Visual Studio - #include directories to Boost

boost/asio.hpp

LINKS:

<https://www.boost.org/doc/libs/1_69_0/doc/html/boost_asio.html>