IRC Chat Program (08March2019)

TODO:

* **DEBUG** logger with levels. Has no constructors, static functions, labeled “**DEBUG**”
* **MAIN** will house a TRY function for **IRC\_PROGRAM** that starts the whole program, and the CATCH will be a crasher for log reports.
* Add debugger outputs for **DEBUG**
* If **DEBUG** is set to 0, do not COUT any debug statements.
* ArgV[1] will be reserved for debugging.
* OPTIONALLY: SSL/SSH? Do not save passwords as plaintext.
* User/Password storage in XML format.
* **SERVER** program created.
* **CLIENT** program created.
* Documentation of the program please.

STRUCTURE:

* **IRC\_PROGRAM** startup, leads into creating the console window for login details such as IP, USER, and PASSWORD. The **CLIENT** will send TCPIP requests to **SERVER** (using bitflags) to create a connection and authenticate.
* **SERVER** should hand off requests to a separate thread each time to handle the request and check for validity. Thread will leave job fail or success so the main thread can then establish and proper connection. Another thread spawned to keep connection alive. When terminated, thread destroyed. Main should not be handling SEND/RECEIVE.
* **CLIENT** will request and authenticate. A separate thread will process SEND/RECEIVE and leave main thread solely for user input and doing SEND requests to child thread.
* **SERVER** will send packets on every chat update. **Client** and **Server** should mirror chat.
* **SERVER** will keep a list of users online and present on the chat.
* **CLIENT** will be able to submit chat commands only visible to the user. **SERVER** will not put commands into the main chat.

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.

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**.

LINKS:

If using a weird structure or code, copy paste website link here.