Notes/Lessons Learned:

1. Update libraries to use windows runtime library/Win32 API
2. Update all console apps to utilize all static libraries and be C through:
   1. Properties->Configuration properties->C/C++->General->add’l include directories
   2. Properties->Configuration properties->VC++ Directories->include directories
   3. Utilize .lib files (in parent directory) in additional dependencies in linker
   4. properties->C/C++->advanced->C compiler
   5. utilize extern C in C++ files, when including C files
3. In the future, make minimal templates for an easier start
4. dir /s \*.lib – lok for .lib fiels recursively
5. WM\_KEYUP is when you let go of a key, WM\_KEYDOWN is pressing it
6. Options for asynchronous:
   1. Hooks ([Hooks Overview - Win32 apps | Microsoft Learn](https://learn.microsoft.com/en-us/windows/win32/winmsg/about-hooks)) with setwindowshookex – only for desktop apps (specifically WH\_KEYBOARD\_LL to catch low level keyboard events) or getmsgproc ([GetMsgProc callback function - Win32 apps | Microsoft Learn](https://learn.microsoft.com/en-us/windows/win32/winmsg/getmsgproc))
   2. Setconsolectrlhandler() -> unknown symbol for cntrl+C
7. Need to do research on symbols and symbol servers for windows
8. Remember to do memory testing on applications in VS (like valgrind): Debug > Performance Profiler
9. Try to get a set up for automatically testing everything – build + run + memory +\_clang-tidy…]
10. Clang-tidy and ASAN set-up:
    1. Clang-tidy: properties->code analysis->enable clang-tidy (built-in VS functionality!)
    2. ASAN:
       1. C/C++->general->enable address sanitizer
       2. C/C++->debug information: change from /ZI to /Zi (/ZI and /fsanitize incompatible)
          1. /ZI enables the edit and continue option, making it incompatible with many VS features
       3. Terminal now requires PATH to asan dlls which are not included in system32 path.
          1. Set in VS: properties->debugging->environment: PATH=$(PATH);C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.41.34120\bin\Hostx64\x64
          2. Set in cmd prompt (for temp cmd prompts): set PATH=C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.41.34120\bin\Hostx64\x64
    3. Could also use CRT heap functions: [Find memory leaks with the CRT library | Microsoft Learn](https://learn.microsoft.com/en-gb/cpp/c-runtime-library/find-memory-leaks-using-the-crt-library?view=msvc-170&viewFallbackFrom=vs-2022)
11. TODO later:
    1. Exception handling
       1. [Chapter 4. Exception Handling | Windows System Programming, Fourth Edition (oreilly.com)](https://learning.oreilly.com/library/view/windows-system-programming/9780321658319/ch04.html)
       2. [Chapter 4. Exception Handling | Windows System Programming, Fourth Edition (oreilly.com)](https://learning.oreilly.com/library/view/windows-system-programming/9780321658319/ch04.html#ch04sec1lev8)
    2. Asynchronous IO and completion ports
       1. [Chapter 14. Asynchronous Input/Output and Completion Ports | Windows System Programming, Fourth Edition (oreilly.com)](https://learning.oreilly.com/library/view/windows-system-programming/9780321658319/ch14.html#ch14)

Notes:

* Look at TCP flow – receiver control

Backlog:

* Hash Table/Linked List Error prints not using thread safe functions
* Switch statement for different packet types within IOCP worker thread
* Create separate packet for sending chat message – i.e. can’t use the same USER packet
* Develop IOCP shutdown – stop all future IO, give timeout to all current IO, wait for total to go to 0, then continue shutdown – use incrementedinterlock etc
* Timestamps on client chats
* Make a compiler option for no console prints: surely this server would go somewhere you don’t want the machine printing to the console. It is a console application though. Maybe some research too.
* Handle E\_OUTOFMEMORY differently
* Handle user list – how to get a list of users to send to the clients
* Handle WSASends when the receiver makes the WSASend wait – need timeout on sends.

Testing:

* MSG > 1018 chars to test user input on client side
* Tested partial sends with smaller packets sizes (setopts SO.SNDBUF)

cd C:\Users\chris\Documents\CSD-GA\TNG\elliott-chat-server\chat\_solution\x64\Debug

set PATH=C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.41.34120\bin\Hostx64\x64