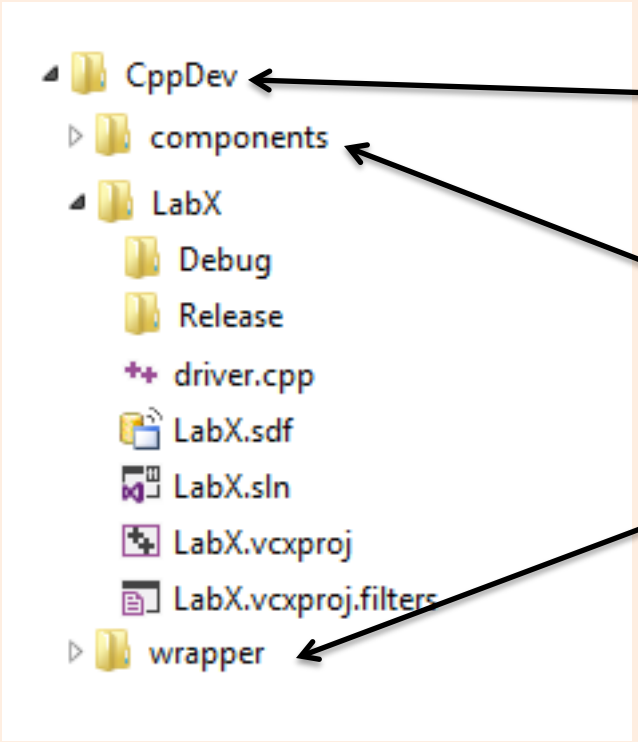
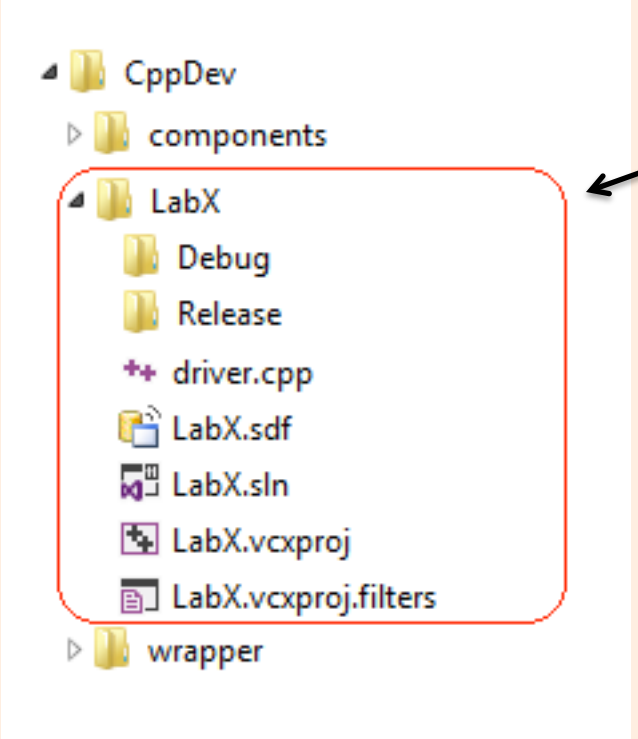
How Visual Studio Compiles a Project

Where Files Reside in File System



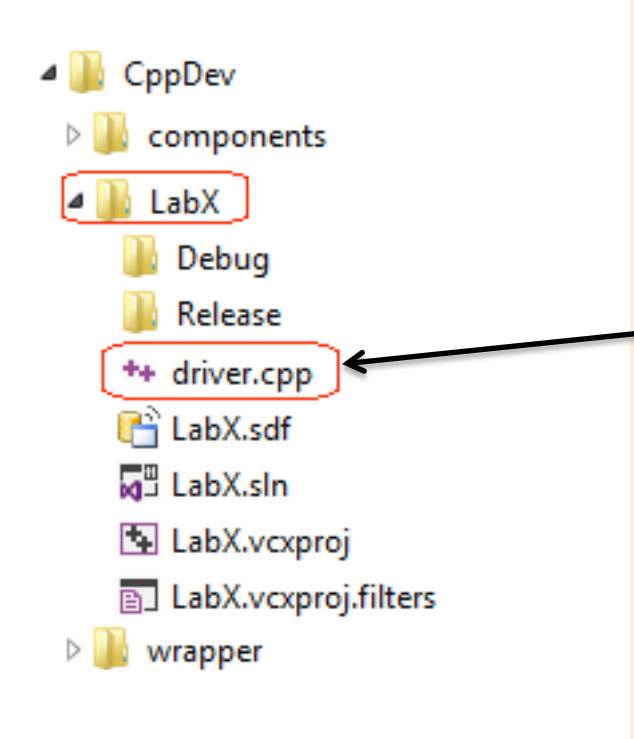
- All files are subordinate to the CppDev folder, which is on your computer's hard driver
- All components, e.g., Queue1 or List1 are in folders subordinate to the components folder
- All the wrapper files are subordinate to the wrapper folder, e.g., wrapper.h

Where Project Files Reside



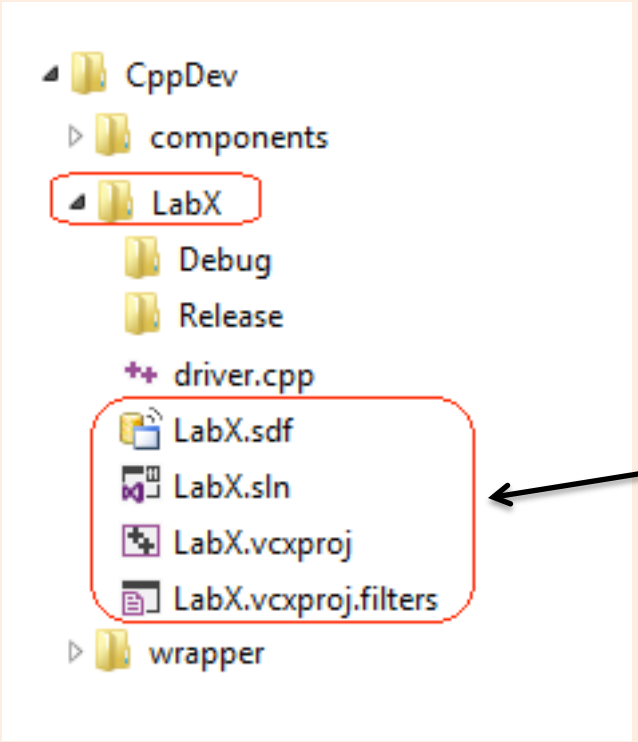
- All labs reside in their own folder, e.g., Lab1 resides in the Lab1 folder

LabX Folder – Source Code



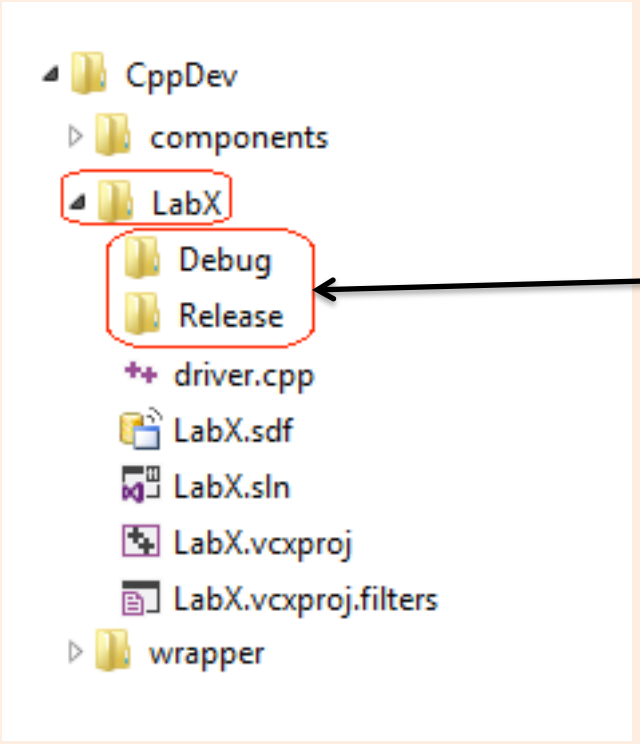
- All code that is *specific* to your project resides in the project's folder, inside the CppDev folder
- In this example, it is the LabX project folder
- Source code typically includes:
  - .cpp files
  - .h files

LabX Folder – Visual Studio Files

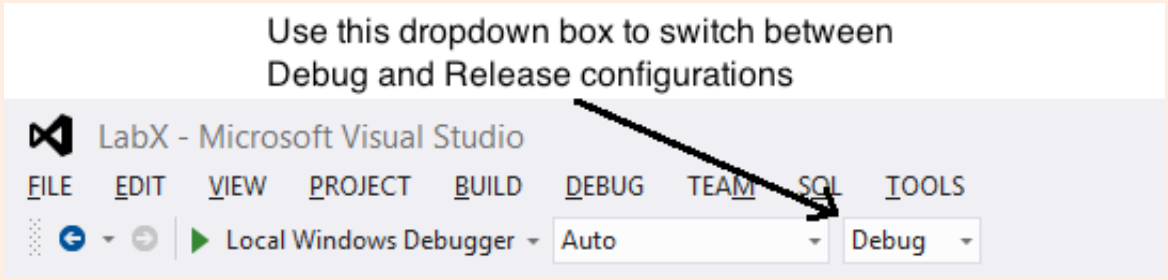


- When you create a Visual Studio project, it creates and maintains a number of project configuration files
- The project in this example is: LabX
- The Visual Studio configuration files created include:
  - .sdf
  - .sln
  - .vcxproj
  - .vcxproj.filters
- From the file system it is possible to double click the .sln file and Visual Studio will open up that project

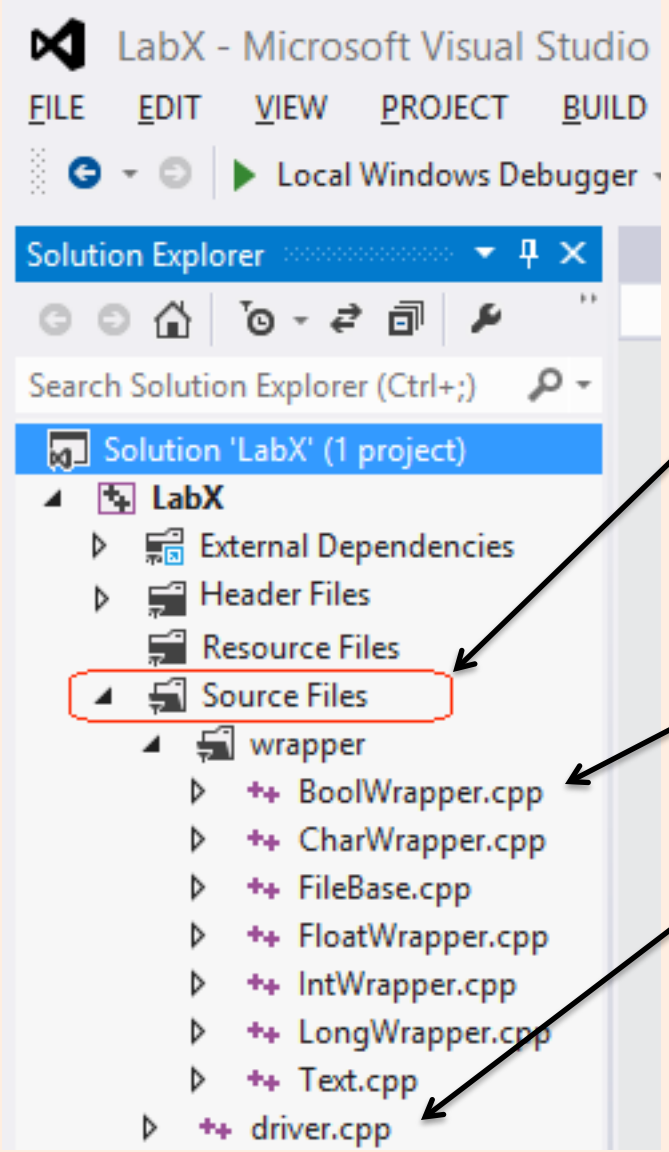
LabX Folder – Subfolders



- Visual Studio creates two subfolders when you build your project
  1. Debug folder – created when you build your project while in *Debug* configuration
  2. Release folder – created when you build your project while in *Release* configuration
- The files that are stored in the Debug & Release folders include:
  - .i – Intermediate file created by the C Preprocessor
  - .obj – Object code files created by the C++ compiler for each .cpp file compiled, e.g., driver.obj for driver.cpp
  - .exe – The executable file created by the Linker, for example LabX.exe for the LabX project



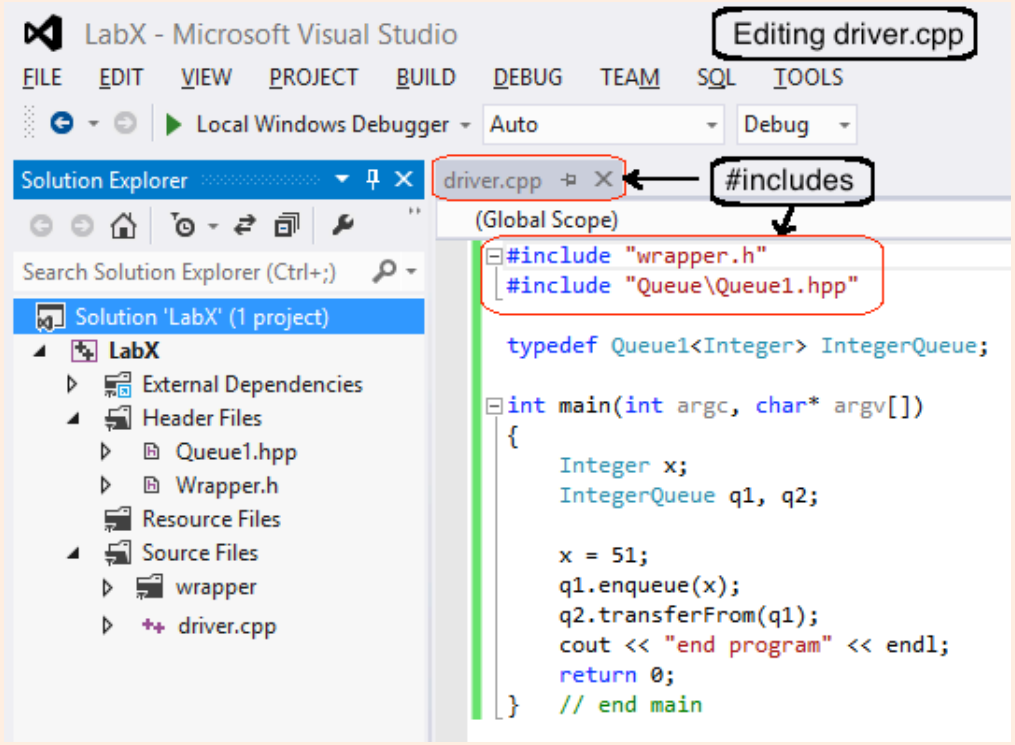
Visual Studio Project



- You must create the Visual Studio project
  - When you *add existing files* to the Source Files folder, you are telling Visual Studio where to find them in the file system, nothing more
  - Most of our projects will need to know where the following source files are located in the file system:
    1. The wrapper .cpp files
    2. Project specific .cpp files

#include Header Files

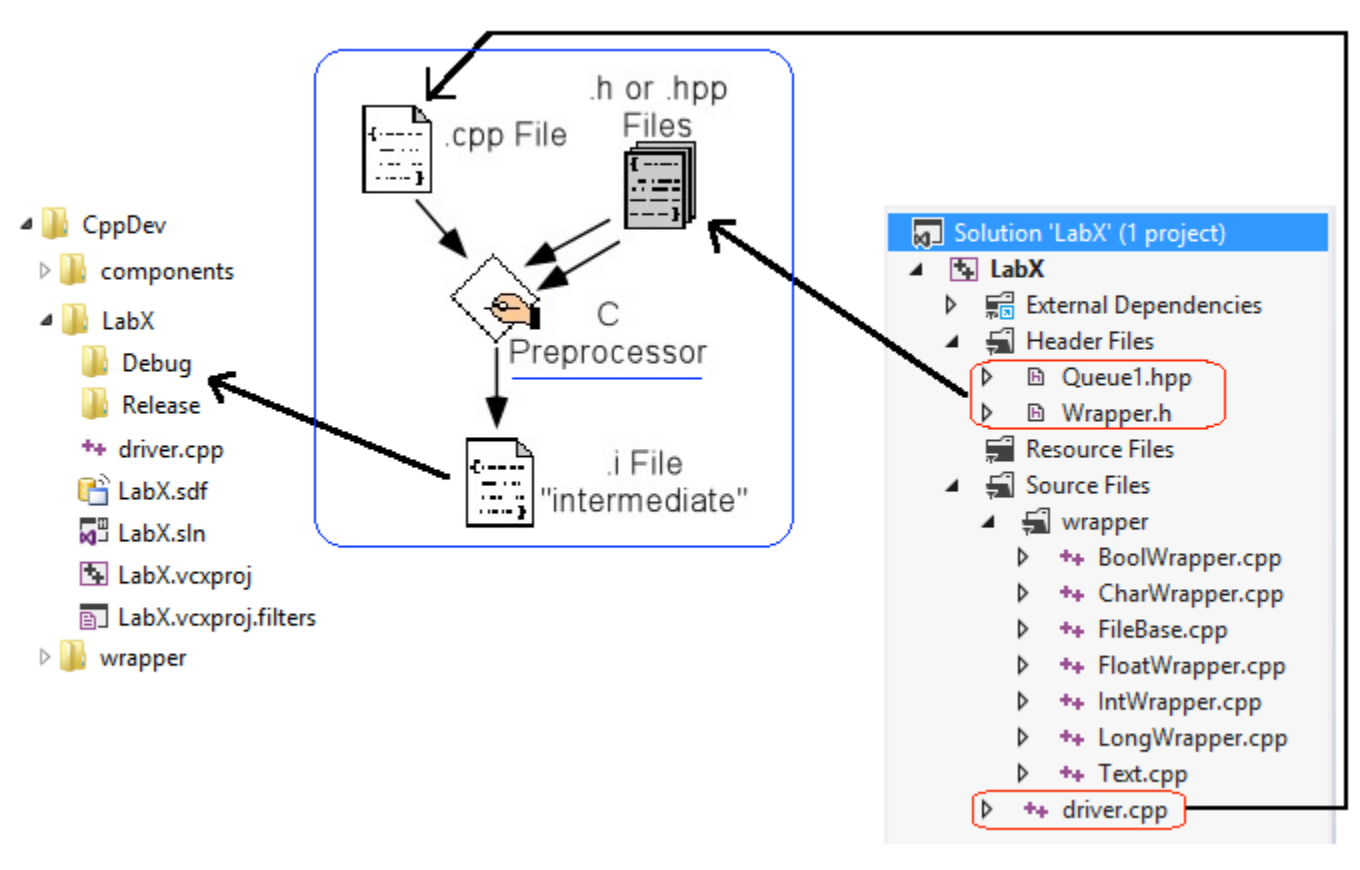
- .cpp files almost always #include header files
- Header files:
  - Have the extensions: .h or .hpp
  - Get the name *header* because they typically contain:
    - Only an operation's signature, i.e., return type, name, and parameter list (found in .h files)
    - Or a template class definition (found in .hpp files)



- In this example, driver.cpp #includes:
  1. "wrapper.h"
  2. "Queue\Queue1.hpp"

The C Preprocessor

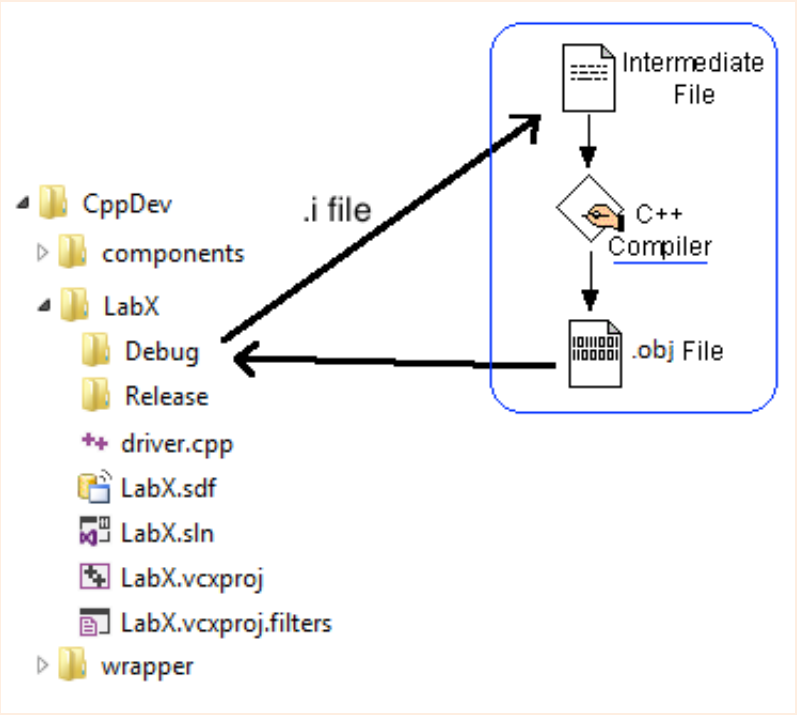
- Comes with all C++ compilers, it processes all the preprocessor directives found in a C++ file
- *preprocessor directives* - begin with a '#' and direct the preprocessor to do various tasks, e.g., to include a file
- The .i file (stands for *intermediate*):
  - Created by the C Preprocessor
  - Contains all the C++ code from the .cpp, with all # preprocessor directives removed (acted upon)
  - Contains the *contents* of all #included files – this is what it means to #include, the contents get included
  - Stored in Debug folder (when in Debug configuration) or Release folder (Release configuration)
  - One .i file is created for each .cpp file found in the project (e.g., driver.i is created from driver.cpp)



C Preprocessor Errors

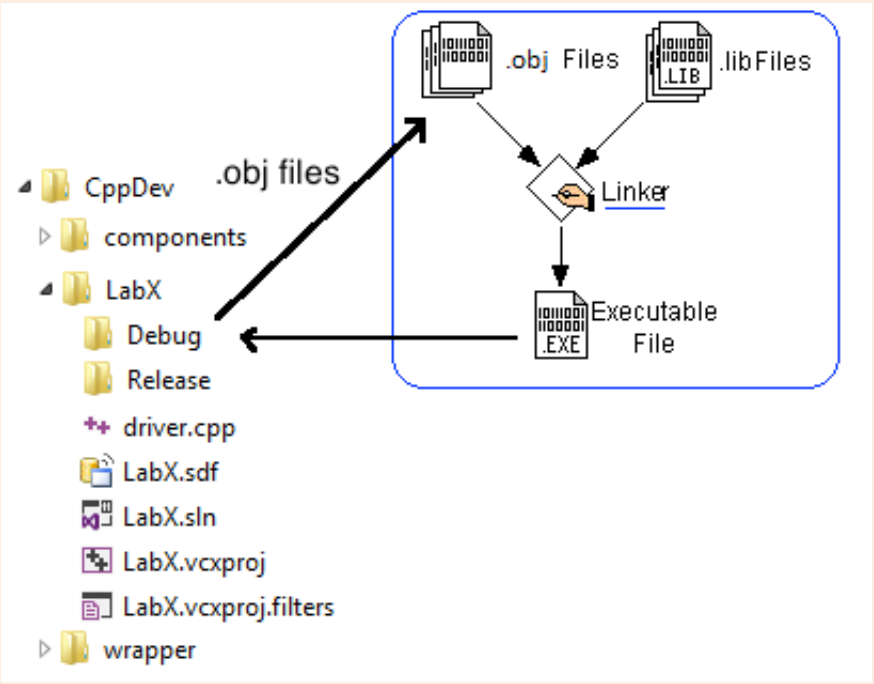
- What can go wrong at this stage?
  - #included files cannot be found
    - They are either not stored in the file system where they are supposed to be
    - Or the #include line is incorrect, e.g., misspelled name, or path name
  - The same file is #included more than once, causing duplication
    - Will cause the compiler to signal an error when it sees some symbol declared more than once
    - Can be stopped by using the #pragma once directive (which is a non-standard directive used by Visual Studio)

C++ Compiler



- For each .i file, the compiler translates its contents into a .obj file
- For example:
  1. driver.i is created by C Preprocessor from driver.cpp
  2. driver.obj is created by C++ compiler from driver.i
- .obj files:
  - Stored in Debug folder (when in Debug configuration) or Release folder (Release configuration)
  - Are not human readable
  - Are also known as an object code file
  - Cannot be executed
- Common compiler errors:
  - Syntax errors, e.g., forgetting semi-colon, misspelling a keyword, etc.
  - Referencing a variable that hasn't been declared
  - Declaring the same variable more than once
  - Mismatched forgetting a close curly brace
  - Etc.

The Linker



The Linker collects all the project's .obj files and the required .lib (Library) files and creates the executable (the .exe)

For example:

1. driver.i is created by C Preprocessor from driver.cpp
2. driver.obj is created by C++ compiler from driver.i
3. driver.exe is created by the Linker

.exe file:

- Stored in Debug folder (when in Debug configuration) or Release folder (Release configuration)
- Are not human readable
- Contains debugging information when built in Debug configuration
- *Debugging information* includes: variable and operation names, parameter names, etc.

- Common Linker errors:
  - When the linker cannot find the compiled code for a called operation – that is, it finds the a call to an operation, but cannot find the actual object code for the called operation
  - When operation's name (and its code) are duplicated in more than one .obj file

Putting It All Together