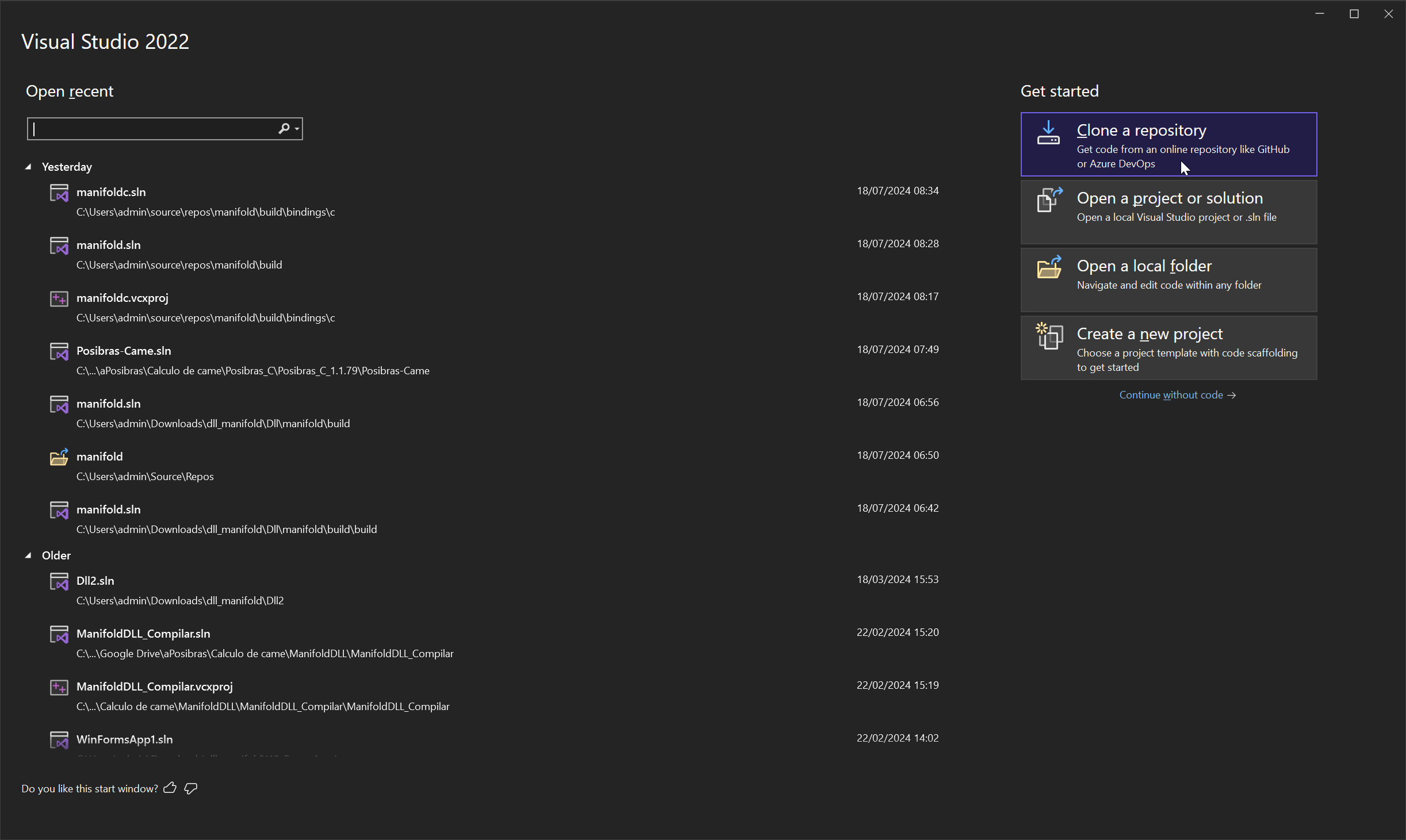
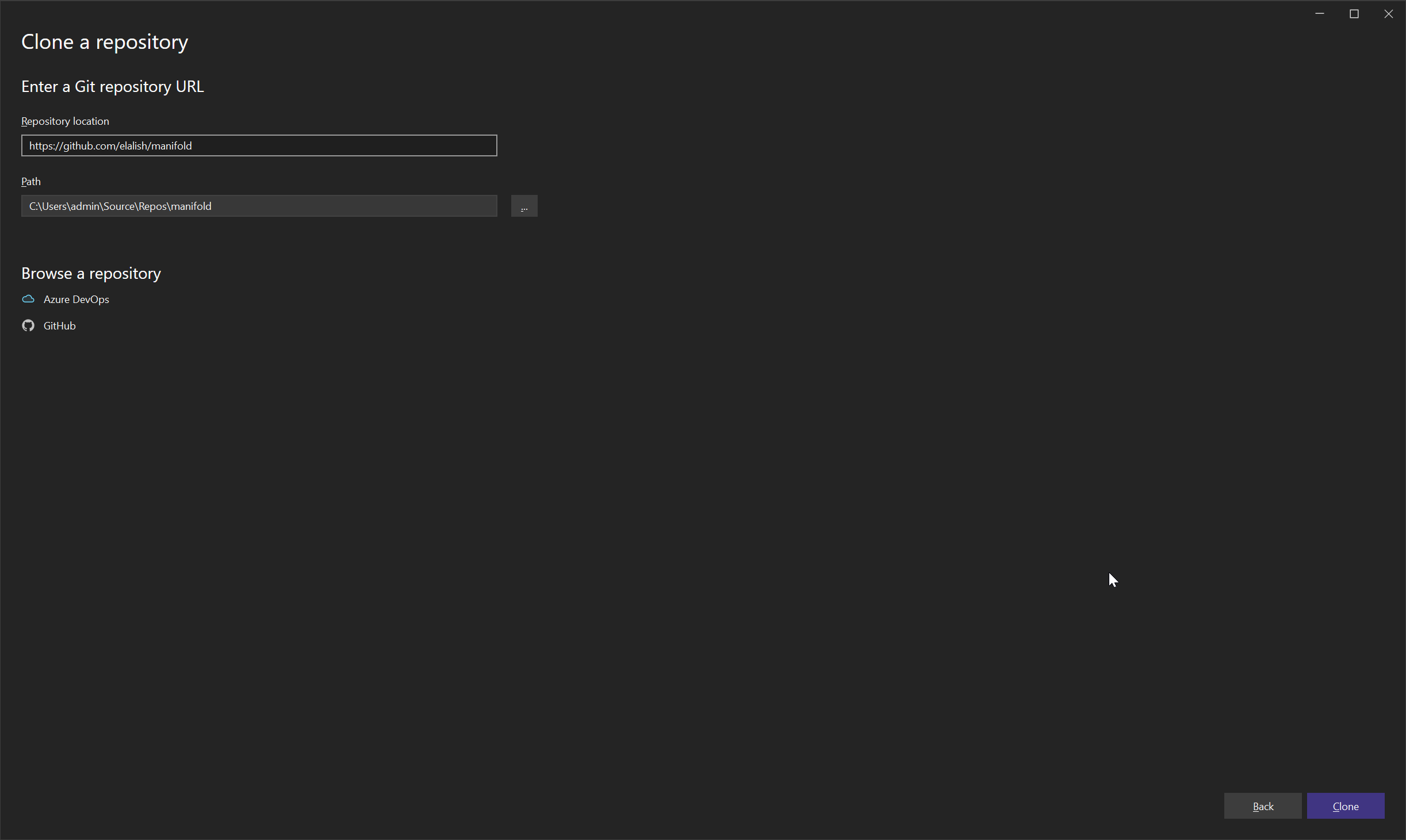
How to compile manifold DLL using VS2022

Open VS2022

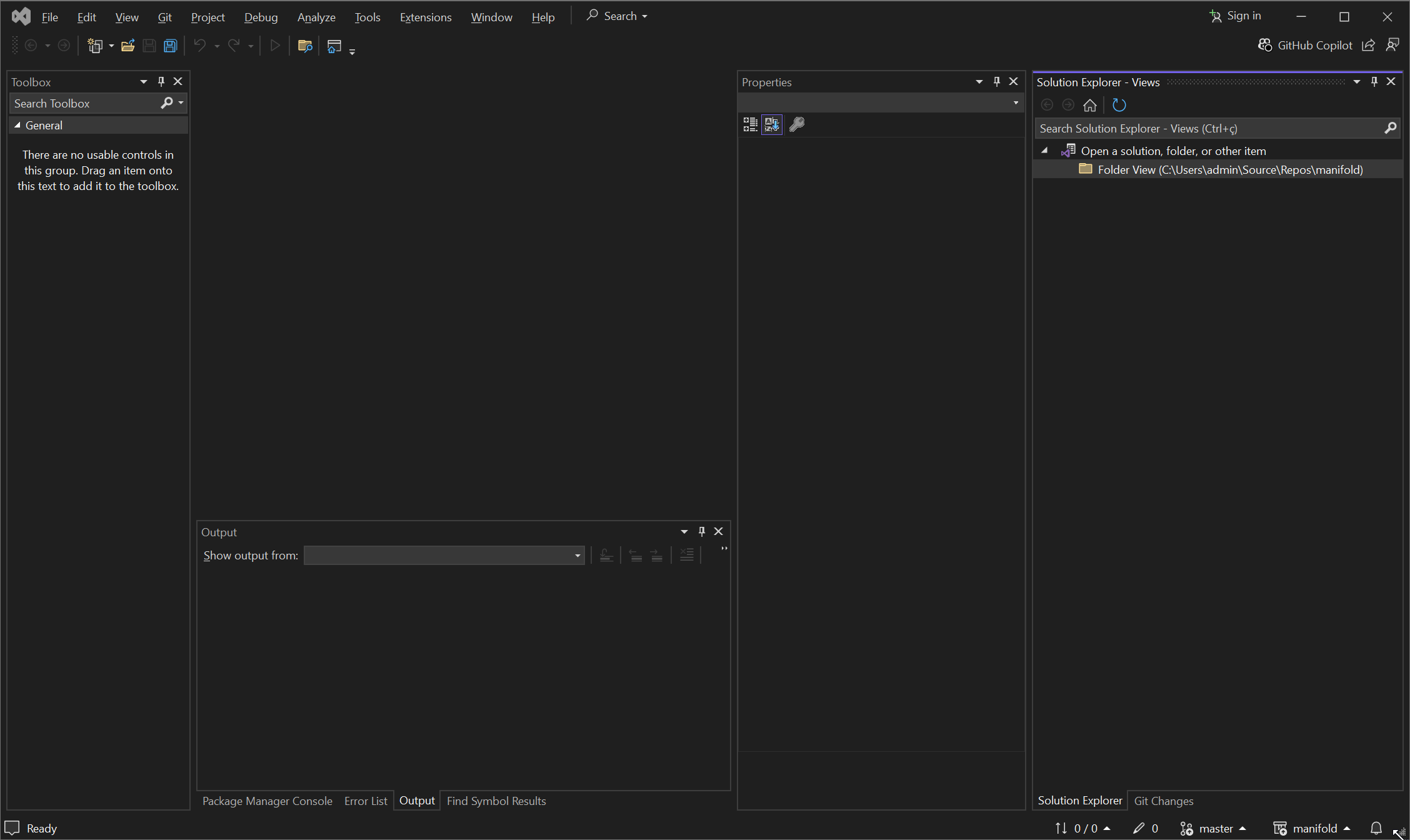
Clone a repository



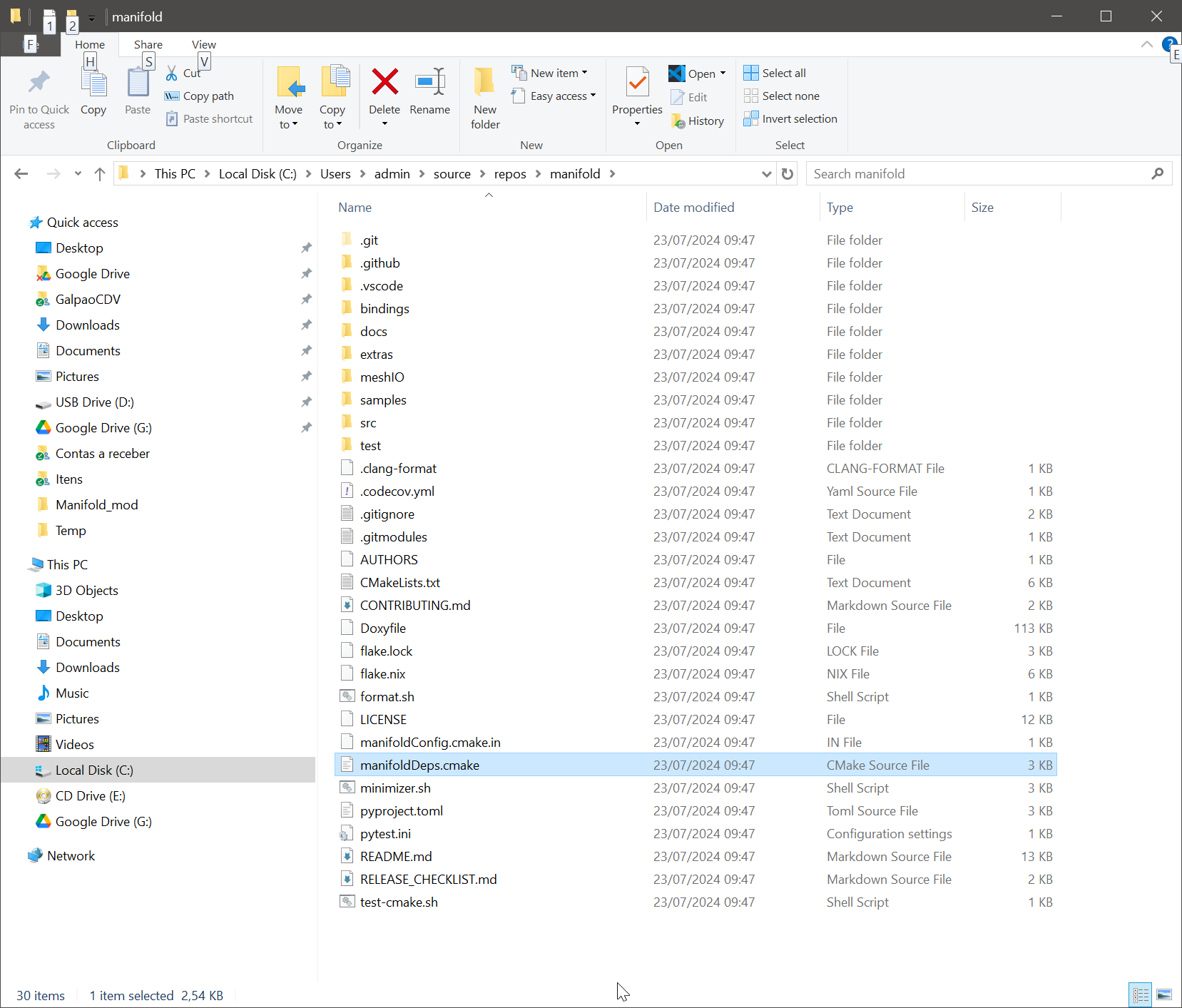
Repository location: <https://github.com/elalish/manifold>



Wait download



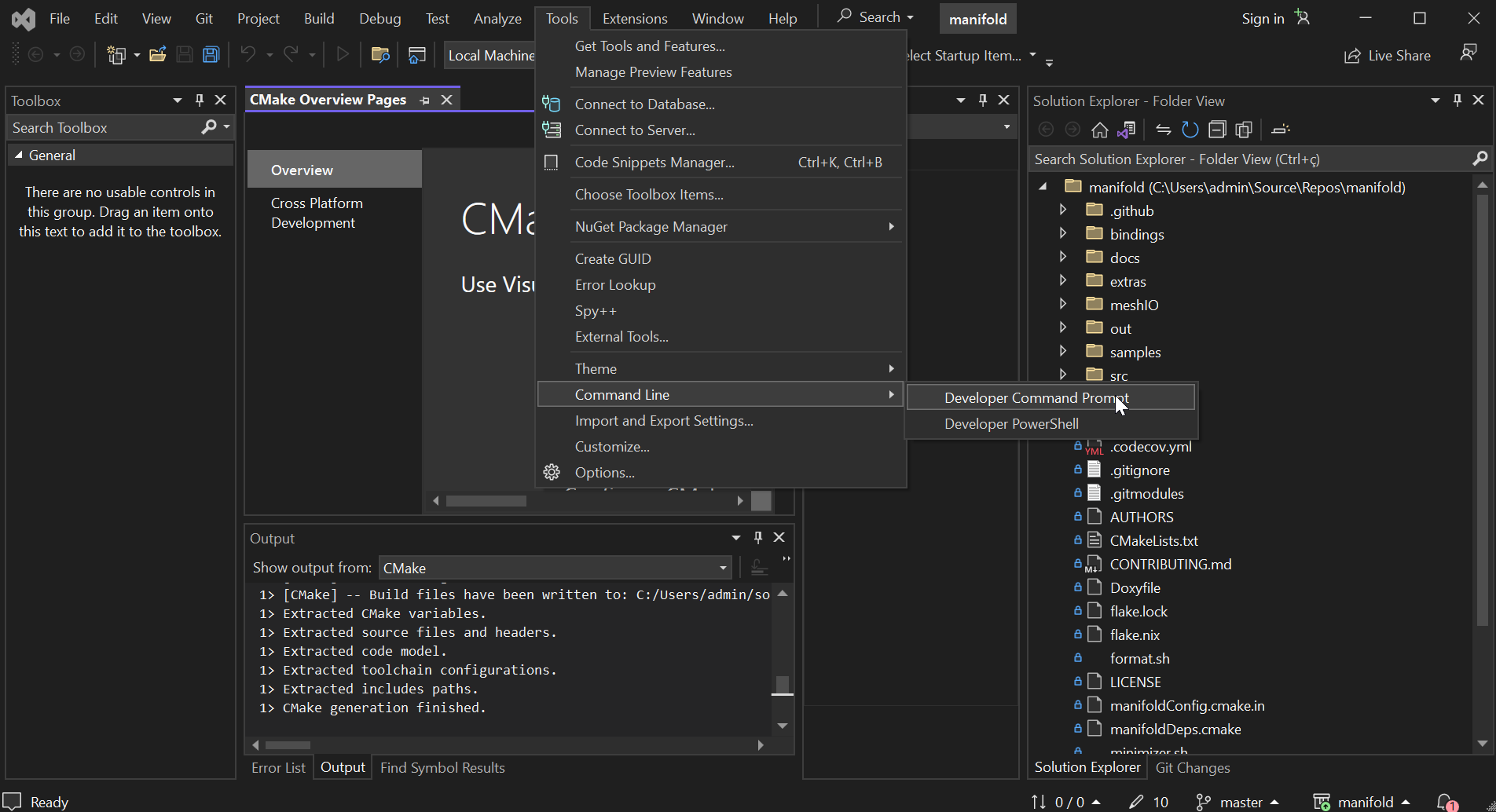
Go to the folder where manifold was installed:



Replace the content of manifoldDeps.cmake to:

|  |
| --- |
| include(FetchContent)  include(GNUInstallDirs)  find\_package(PkgConfig QUIET)  find\_package(Clipper2 QUIET)  if(MANIFOLD\_PAR STREQUAL "TBB")  find\_package(TBB QUIET)  if(APPLE)  find\_package(oneDPL QUIET)  endif()  endif()  if (PKG\_CONFIG\_FOUND)  if (NOT Clipper2\_FOUND)  pkg\_check\_modules(Clipper2 Clipper2)  endif()  if(MANIFOLD\_PAR STREQUAL "TBB" AND NOT TBB\_FOUND)  pkg\_check\_modules(TBB tbb)  endif()  endif()  if(Clipper2\_FOUND)  add\_library(Clipper2 SHARED IMPORTED)  set\_property(TARGET Clipper2 PROPERTY  IMPORTED\_LOCATION ${Clipper2\_LINK\_LIBRARIES})  if(WIN32)  set\_property(TARGET Clipper2 PROPERTY  IMPORTED\_IMPLIB ${Clipper2\_LINK\_LIBRARIES})  endif()  target\_include\_directories(Clipper2 INTERFACE ${Clipper2\_INCLUDE\_DIRS})  else()  message(STATUS "clipper2 not found, downloading from source")  set(CLIPPER2\_UTILS OFF)  set(CLIPPER2\_EXAMPLES OFF)  set(CLIPPER2\_TESTS OFF)  set(CLIPPER2\_USINGZ "OFF" CACHE STRING "Preempt cache default of USINGZ (we only use 2d)")  FetchContent\_Declare(Clipper2  GIT\_REPOSITORY https://github.com/AngusJohnson/Clipper2.git  GIT\_TAG ff378668baae3570e9d8070aa9eb339bdd5a6aba  GIT\_PROGRESS TRUE  SOURCE\_SUBDIR CPP  )  FetchContent\_MakeAvailable(Clipper2)  if(NOT EMSCRIPTEN)  install(TARGETS Clipper2)  endif()  endif()  find\_package(glm QUIET)  if(NOT glm\_FOUND)  message(STATUS "glm not found, downloading from source")  set(GLM\_BUILD\_INSTALL "ON" CACHE STRING "")  FetchContent\_Declare(glm  GIT\_REPOSITORY https://github.com/g-truc/glm.git  GIT\_TAG 1.0.1  GIT\_PROGRESS TRUE  )  FetchContent\_MakeAvailable(glm)  if(NOT EMSCRIPTEN)  install(TARGETS glm)  endif()  endif()  if(MANIFOLD\_PAR STREQUAL "TBB" AND NOT TBB\_FOUND)  message(STATUS "tbb not found, downloading from source")  include(FetchContent)  set(TBB\_TEST OFF CACHE INTERNAL "" FORCE)  set(TBB\_STRICT OFF CACHE INTERNAL "" FORCE)  FetchContent\_Declare(TBB  GIT\_REPOSITORY https://github.com/oneapi-src/oneTBB.git  GIT\_TAG v2021.11.0  GIT\_PROGRESS TRUE  )  FetchContent\_MakeAvailable(TBB)  set\_property(DIRECTORY ${tbb\_SOURCE\_DIR} PROPERTY EXCLUDE\_FROM\_ALL YES)  if(NOT EMSCRIPTEN)  install(TARGETS tbb)  endif()  endif()  if(MANIFOLD\_EXPORT)  find\_package(assimp QUIET)  if(NOT assimp\_FOUND)  message(STATUS "assimp not found, downloading from source")  FetchContent\_Declare(assimp  GIT\_REPOSITORY https://github.com/assimp/assimp.git  GIT\_TAG v5.2.5  GIT\_PROGRESS TRUE  )  FetchContent\_MakeAvailable(assimp)  if(NOT TARGET assimp)  add\_library(assimp SHARED IMPORTED)  set\_property(TARGET assimp PROPERTY  IMPORTED\_LOCATION ${assimp\_LINK\_LIBRARIES})  if(WIN32)  set\_property(TARGET assimp PROPERTY  IMPORTED\_IMPLIB ${assimp\_LINK\_LIBRARIES})  endif()  target\_include\_directories(assimp INTERFACE ${assimp\_INCLUDE\_DIRS})  if(NOT EMSCRIPTEN)  install(TARGETS assimp)  endif()  endif()  endif()  endif() |

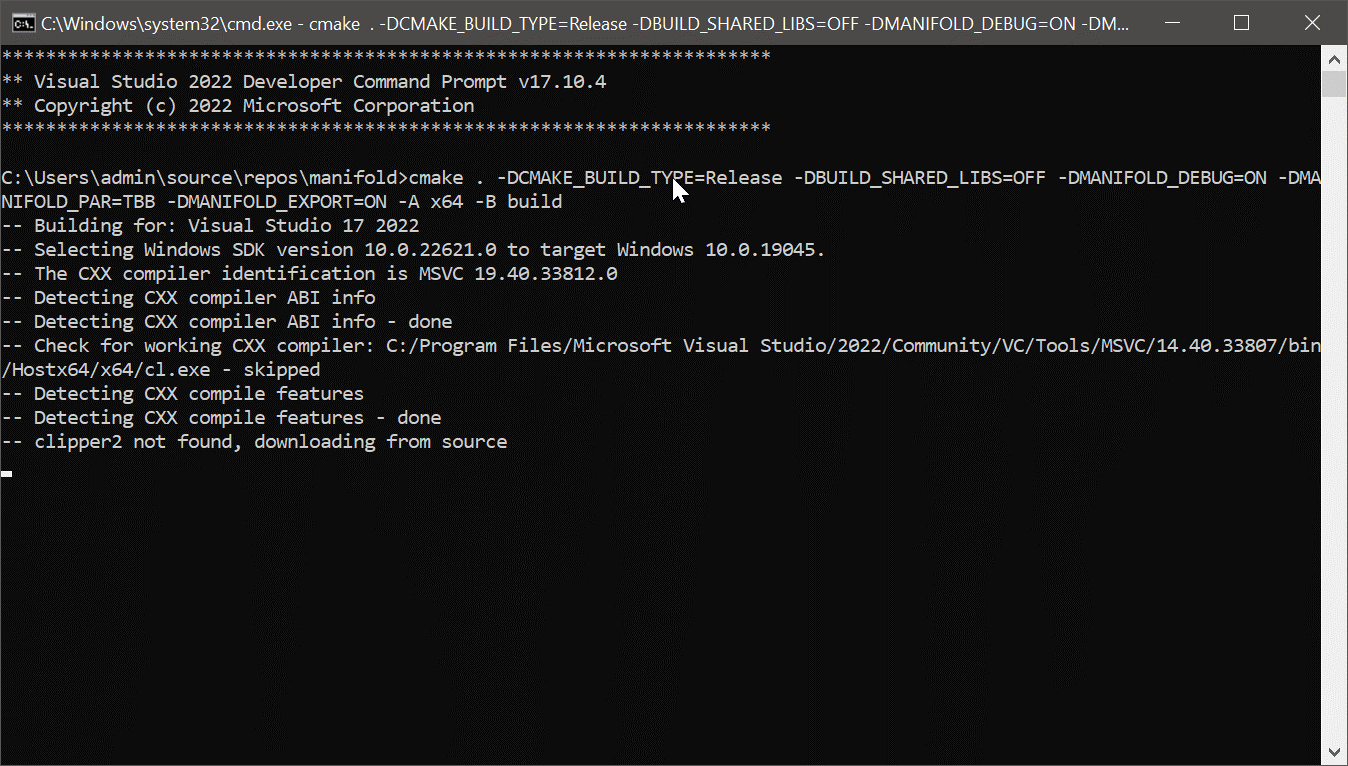
IN VS2022, click on Tool / Command Line / Developer Command Prompt



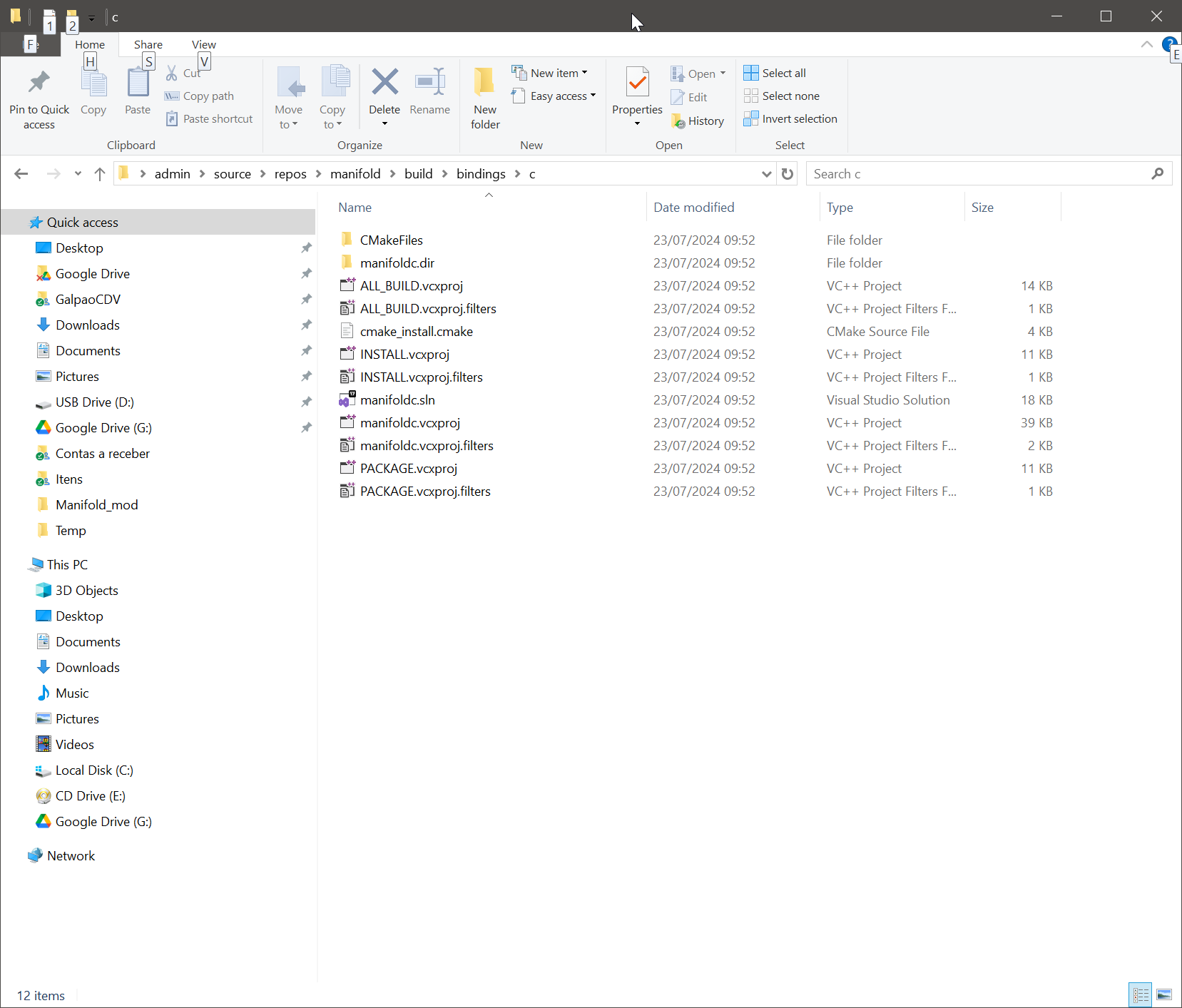
It will open a command prompt.

Run:

cmake . -DCMAKE\_BUILD\_TYPE=Release -DBUILD\_SHARED\_LIBS=OFF -DMANIFOLD\_DEBUG=ON -DMANIFOLD\_PAR=TBB -DMANIFOLD\_EXPORT=ON -A x64 -B build

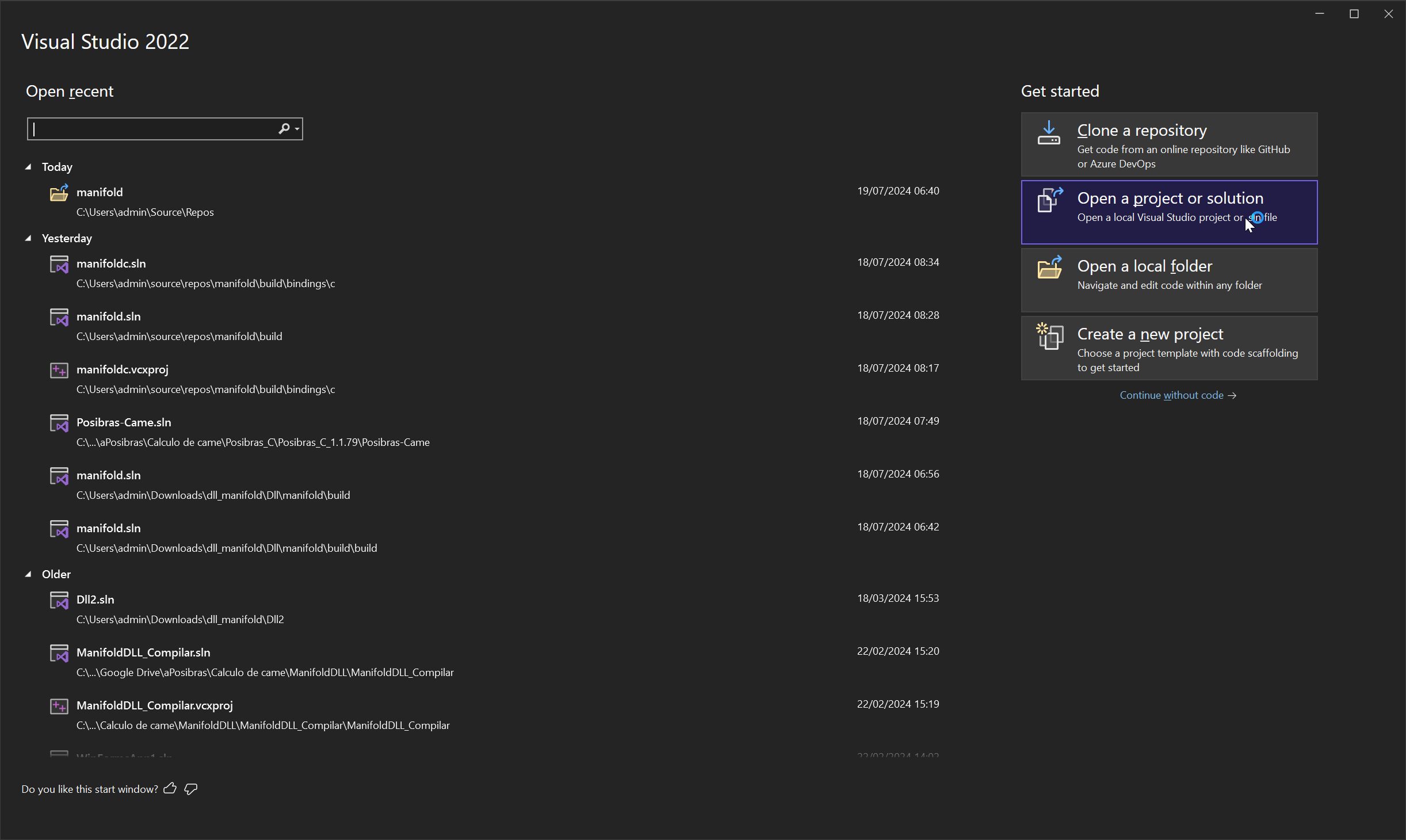


The step above will create a folder in manifold\build\bindings\c

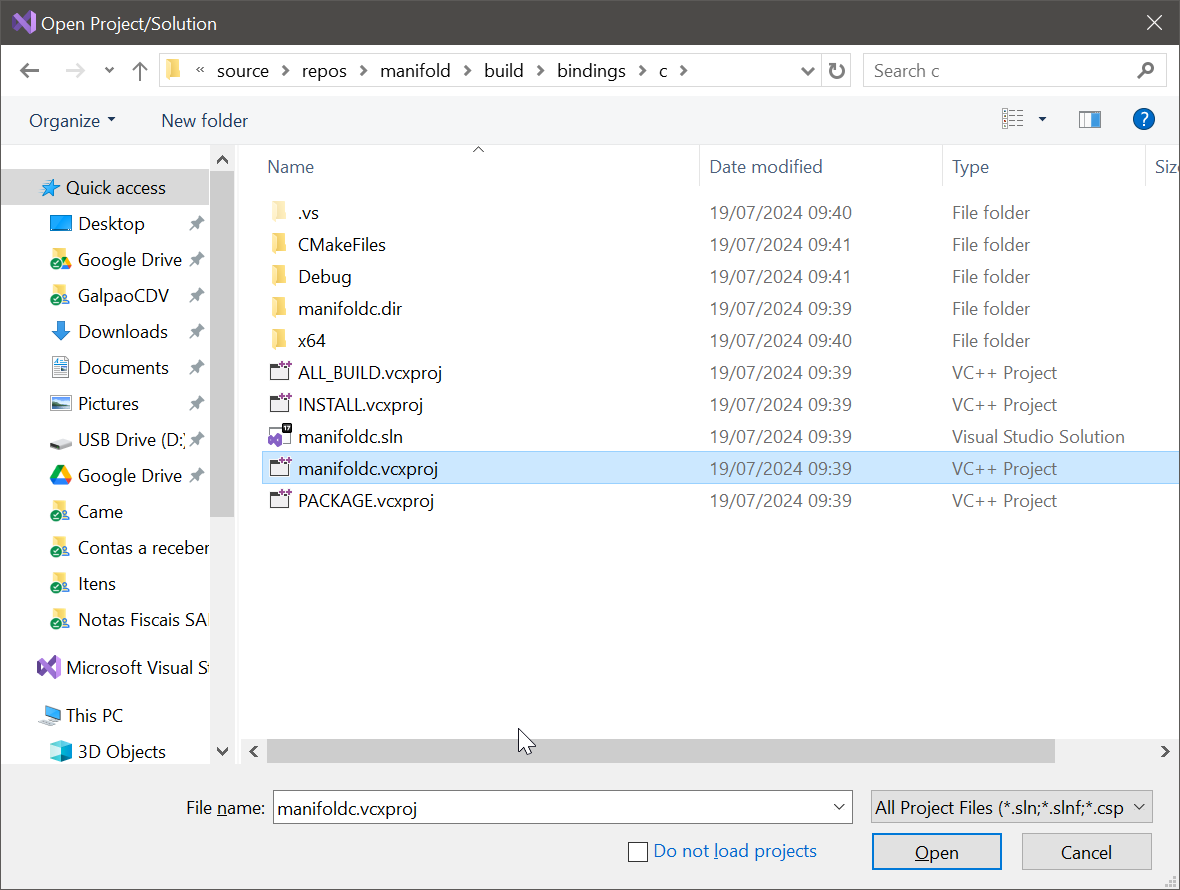


Close the Command prompt screen and also close VS2022

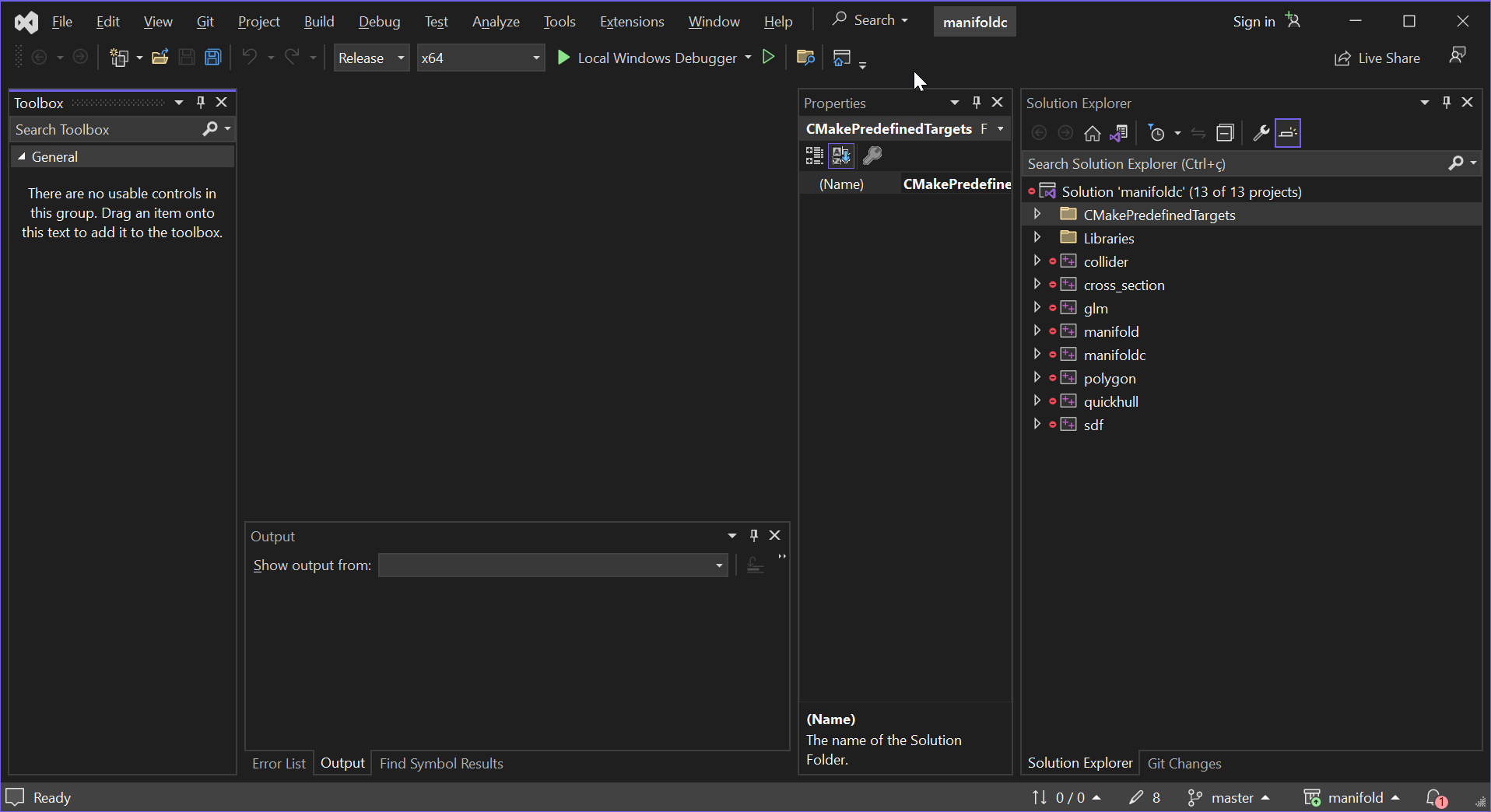
Open a new VS2022 anc click on Open a project or solution



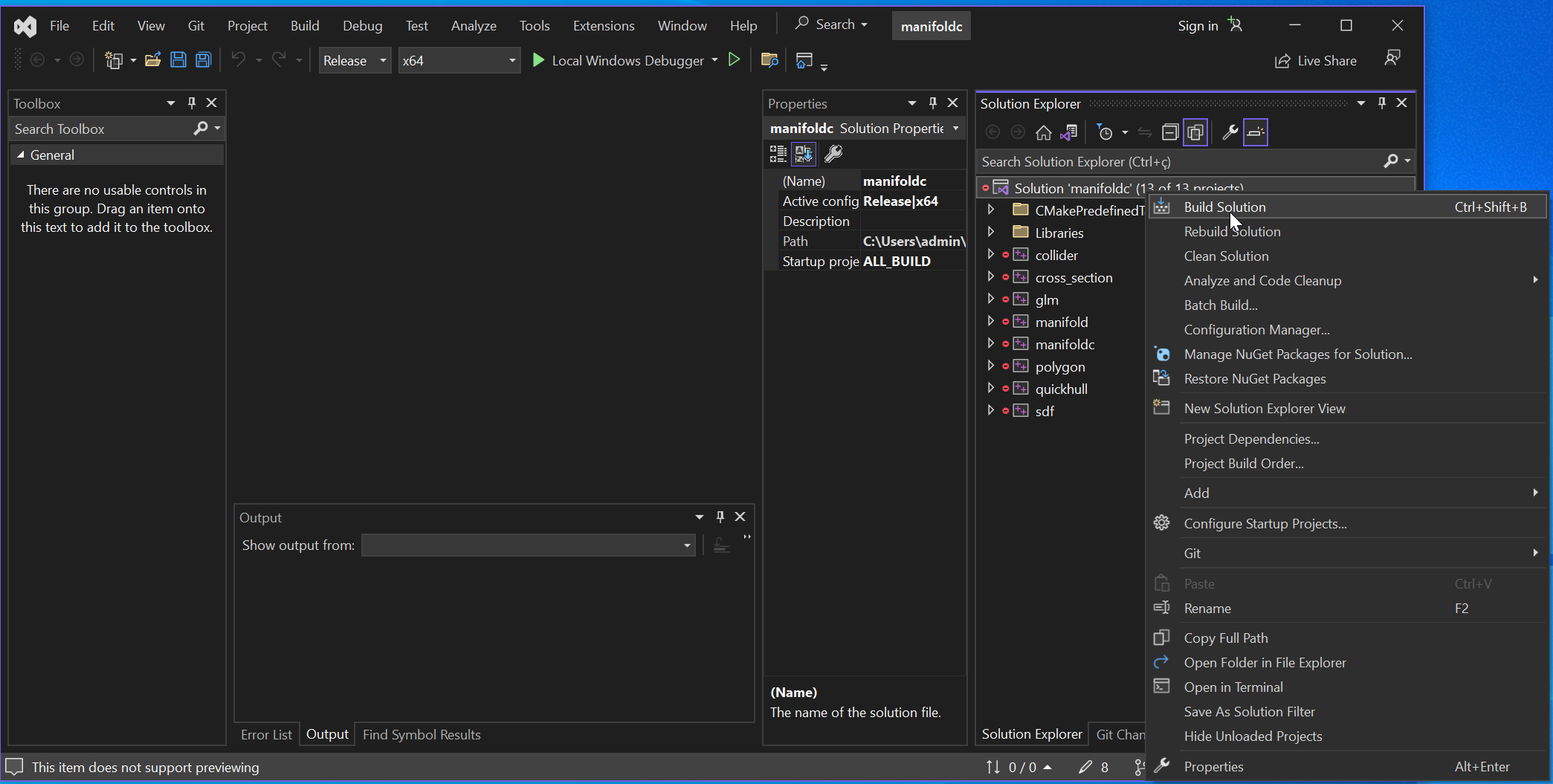
**Open the file manifoldc.vcxproj**



Select Release or Degub



Right click on Solution ‘manifoldc’ and click on Build Solution



The DLL will be created on

manifold\build\bin\Release\manifoldc.dll (Release version)

or

manifold\build\bin\Debug\manifoldcd.dll (Debub version)

Sample program in C#

|  |
| --- |
| // See https://aka.ms/new-console-template for more information  using System;  using System.Runtime.InteropServices;  using System.Text;  class Program  {  static void Main(string[] args)  {  const string DllPath = @"C:\Users\admin\manifold\build\bin\Release\manifoldc.dll ";  [DllImport(DllPath, CallingConvention = CallingConvention.Cdecl)]  static extern nuint manifold\_cylinder(nuint mem, float height, float radiusLow, float radiusHigh, int circularSegments, int center);  [DllImport(DllPath, CallingConvention = CallingConvention.Cdecl)]  static extern nuint manifold\_get\_meshgl(nuint mem, nuint manifold\_ptr);  [DllImport(DllPath, CallingConvention = CallingConvention.Cdecl)]  static extern void manifold\_export\_meshgl([MarshalAs(UnmanagedType.LPStr)] string file\_path, nuint mesh\_ptr,nuint export\_options\_ptr);  [DllImport(DllPath, CallingConvention = CallingConvention.Cdecl)]  static extern nuint manifold\_export\_options(nuint mem);  nint addr\_export\_opt = Marshal.AllocHGlobal(96); // Size of class ExportOptions: 96 bytes  addr\_export\_opt = (nint)manifold\_export\_options((nuint)addr\_export\_opt);  nint addr = Marshal.AllocHGlobal(16); // Size of class manifold: 16 bytes  addr = (nint) manifold\_cylinder((nuint)addr, 10.0f, 5.0f, 5.0f, 360, 1);  nint addr\_mesh = Marshal.AllocHGlobal(304); // Size of class Mesh: 304 bytes  addr\_mesh = (nint)manifold\_get\_meshgl((nuint)addr\_mesh,(nuint)addr);    string file = "test.glb";    manifold\_export\_meshgl(file, (nuint)addr\_mesh, (nuint)addr\_export\_opt);  return;  }  } |