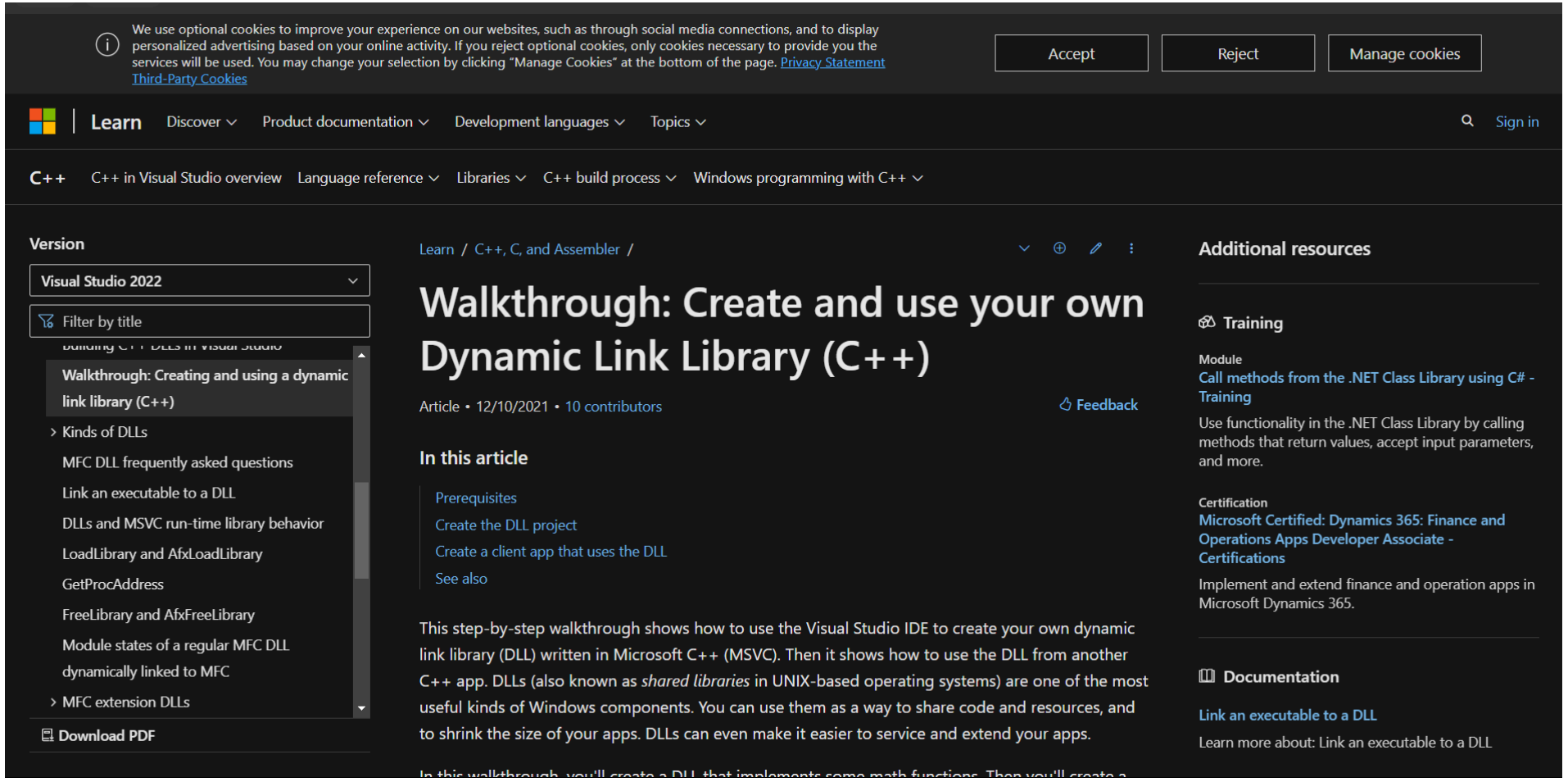
Guide to Dll

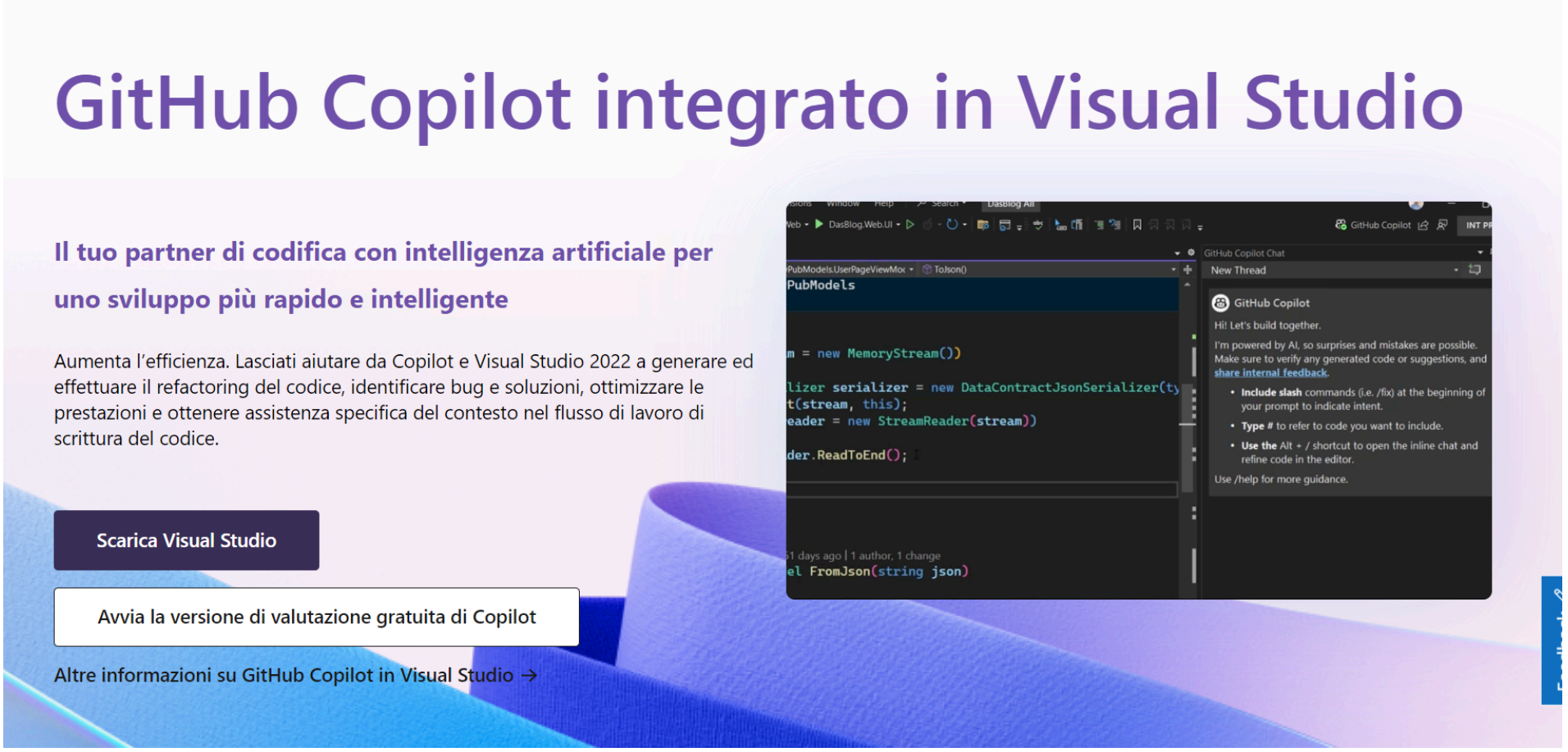
1. Official Microsoft guide: https://learn.microsoft.com/en-us/cpp/build/walkthrough-creating-and-using-a-dynamic-link-library-cpp?view=msvc-170

- Reference:



2. Visual Studio: https://visualstudio.microsoft.com/it/

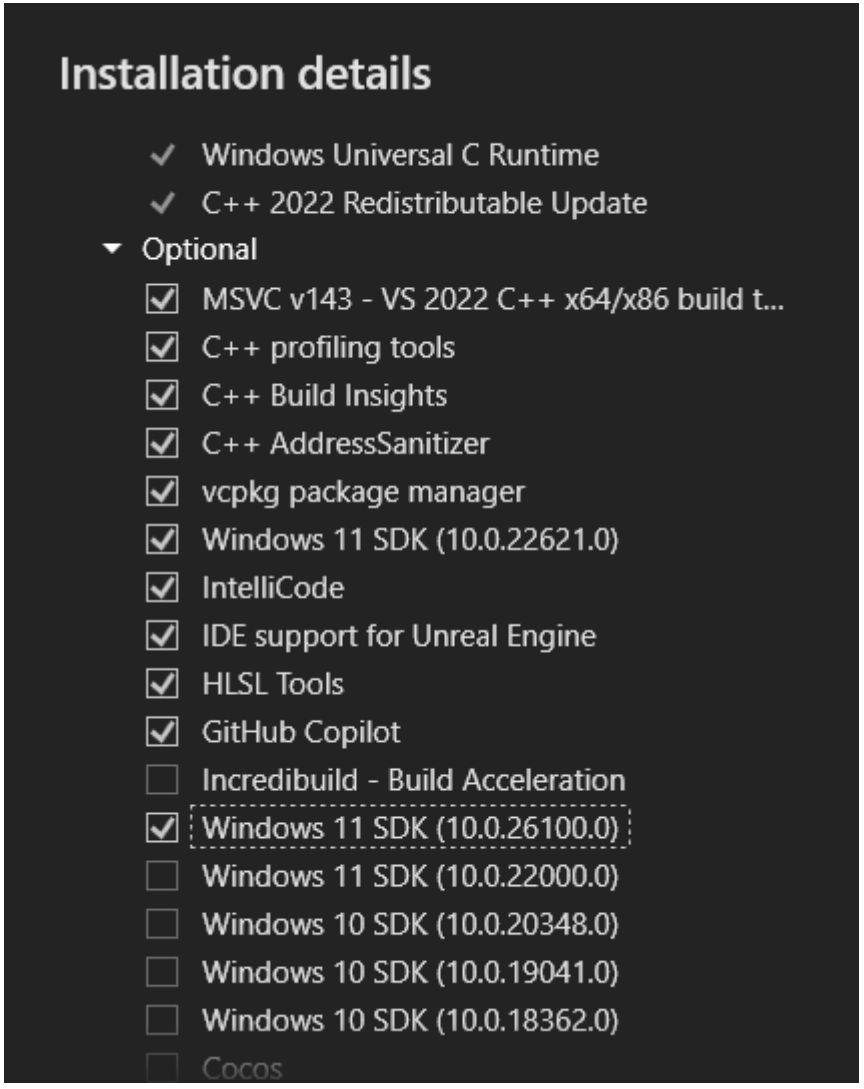
- Download link: https://visualstudio.microsoft.com/it/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&cid=2030&passive=false
- Reference:



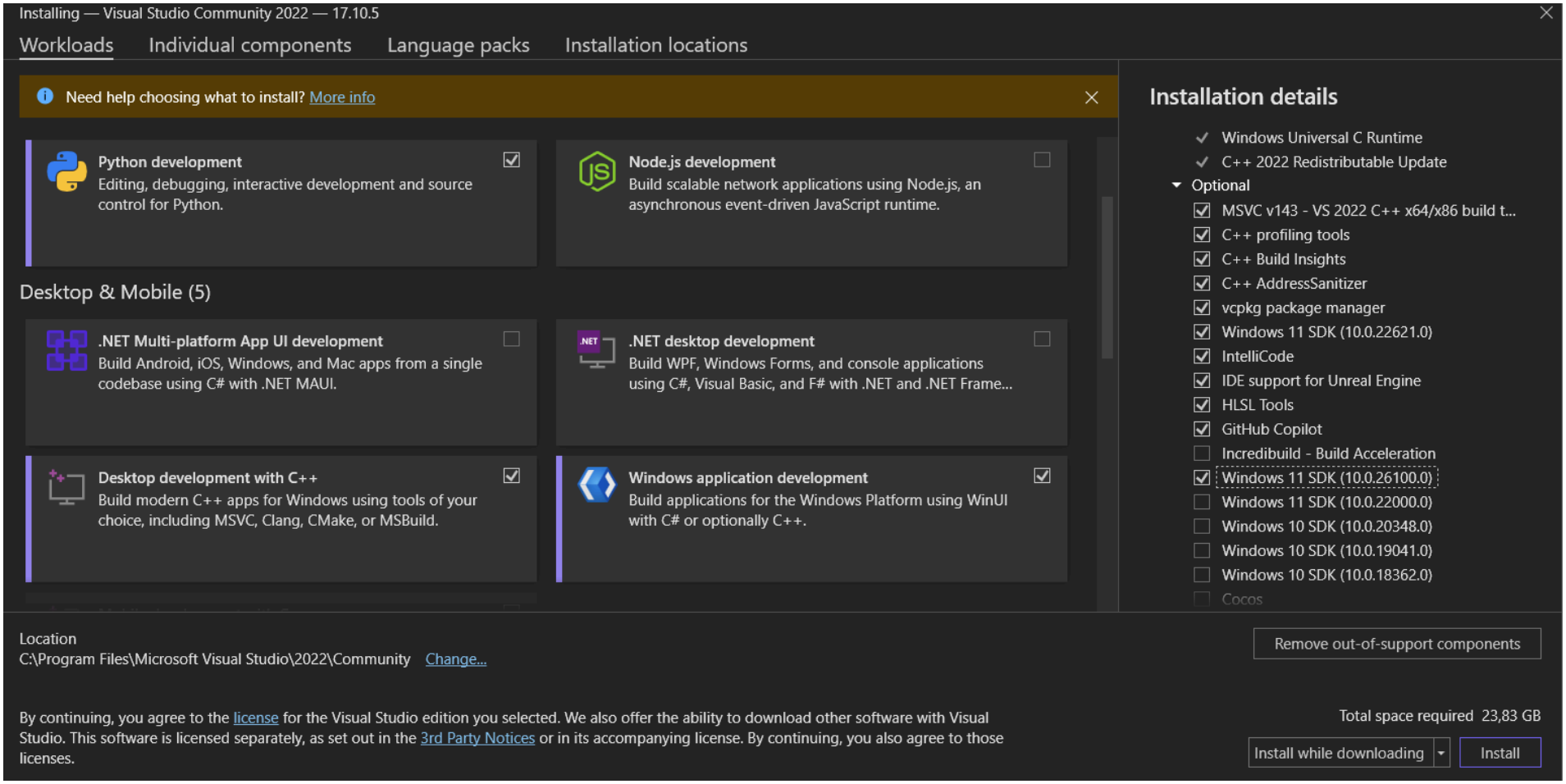
---

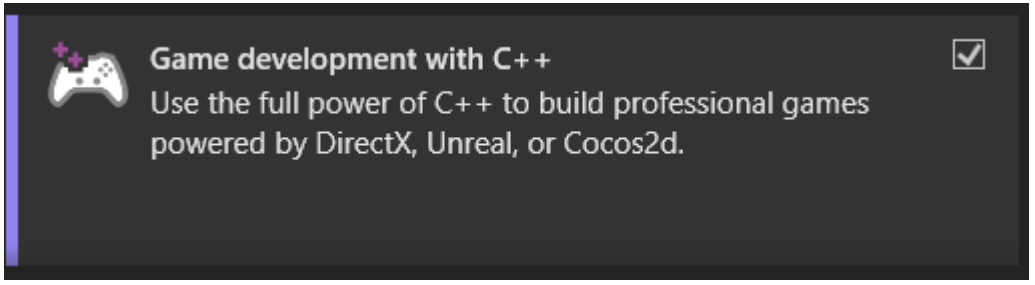
3. On Visual Studio:

- Required:
  `Python development, Game development with C++, Windows application development, Desktop development with C++`
- Installation details :

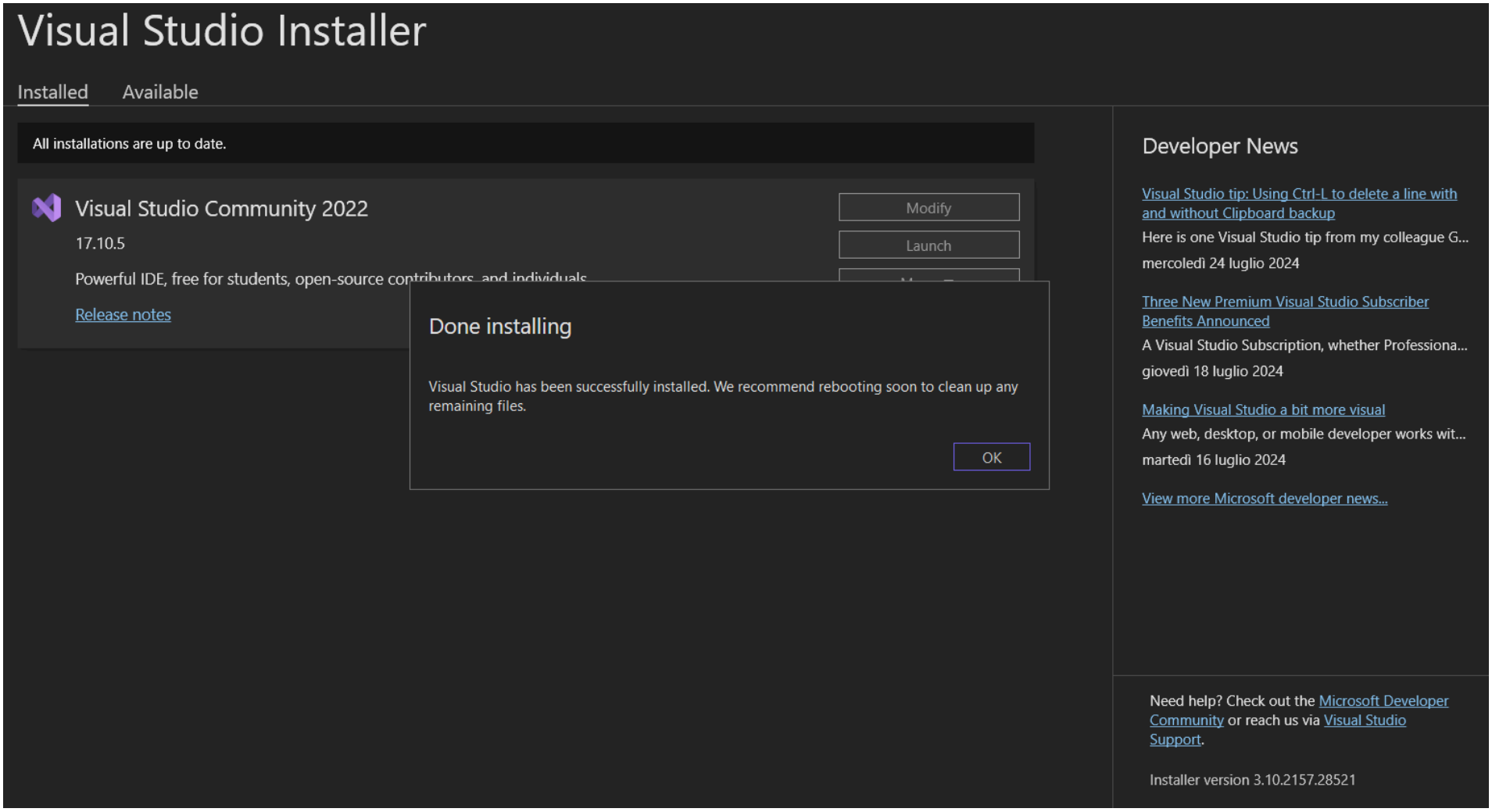


- Reference:


Installing — Visual Studio Community 2022 — 17.10.5
Workloads
Individual components
Language packs
Installation locations
Need help choosing what to install? More info
Python development
Editing, debugging, interactive development and source control for Python.
Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.
Desktop & Mobile (5)
.NET Multi-platform App UI development
Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.
.NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
Windows application development
Build applications for the Windows Platform using WinUI with C# or optionally C++.
Installation details
Windows Universal C Runtime
C++ 2022 Redistributable Update
Optional
MSVC v143 - VS 2022 C++ x64/x86 build t...
C++ profiling tools
C++ Build Insights
C++ AddressSanitizer
vcpkg package manager
Windows 11 SDK (10.0.22621.0)
IntelliCode
IDE support for Unreal Engine
HLSL Tools
GitHub Copilot
Incredibuild - Build Acceleration
Windows 11 SDK (10.0.26100.0)
Windows 11 SDK (10.0.22000.0)
Windows 10 SDK (10.0.20348.0)
Windows 10 SDK (10.0.19041.0)
Windows 10 SDK (10.0.18362.0)
Cocos
Location
C:\Program Files\Microsoft Visual Studio\2022\Community
Change...
Remove out-of-support components
By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.
Total space required 23,83 GB
Install while downloading
Install


Game development with C++
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

- Click "install" -> so click "ok" when it is done:

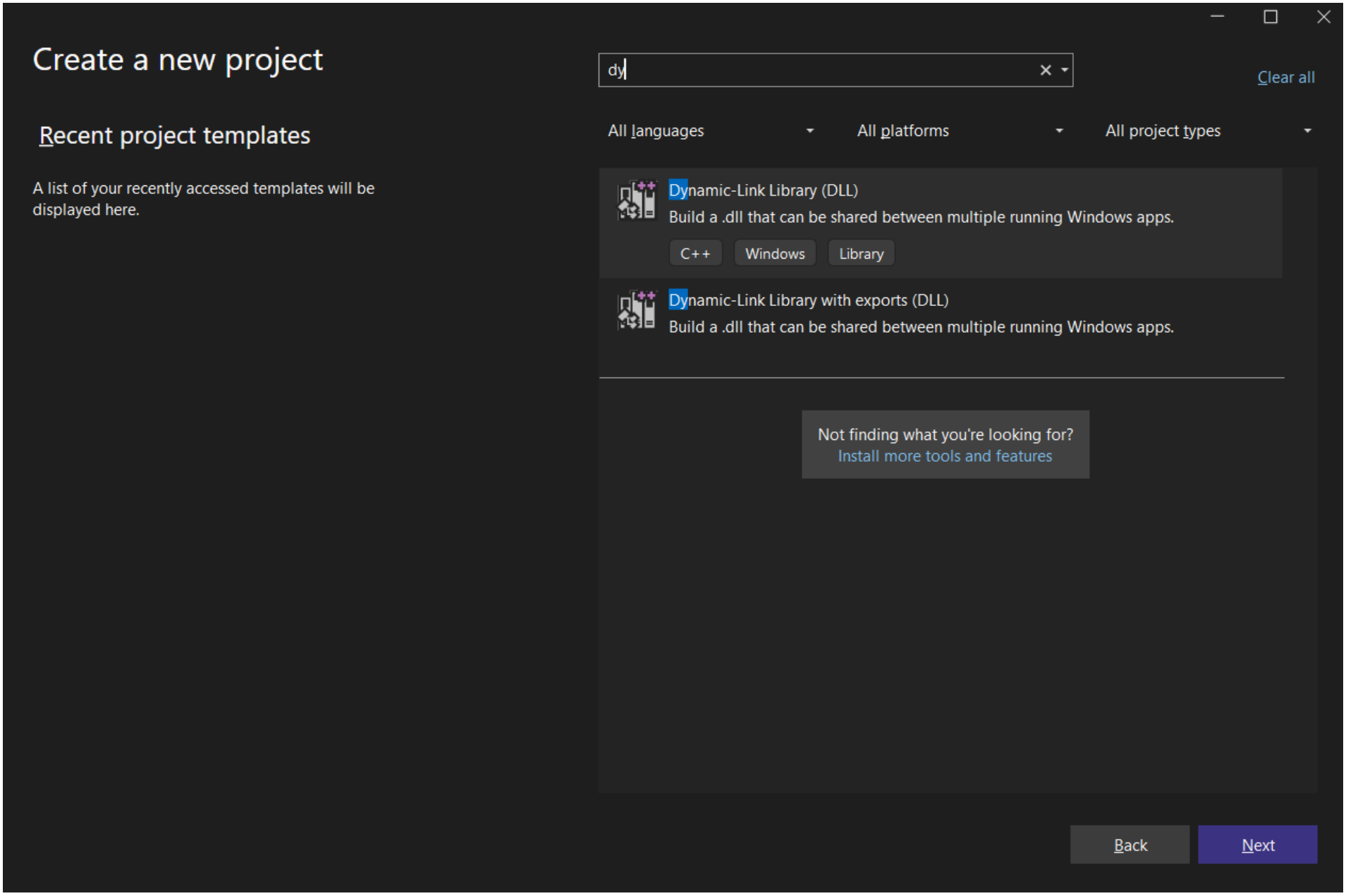


---

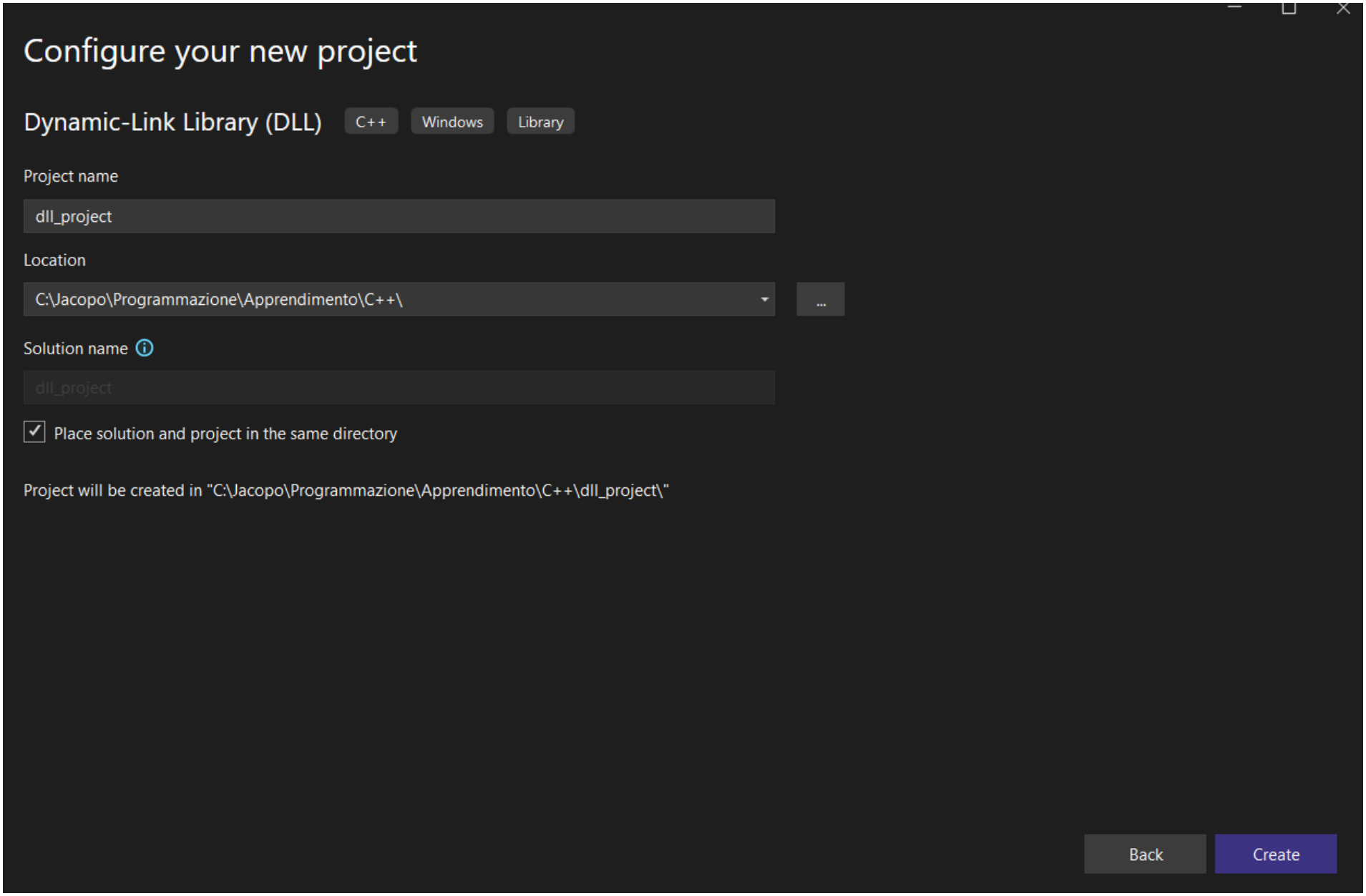
4. Create a project:

- Click on "create a new project" into Visual Studio. (I'm using Visual Studio Community 2022).

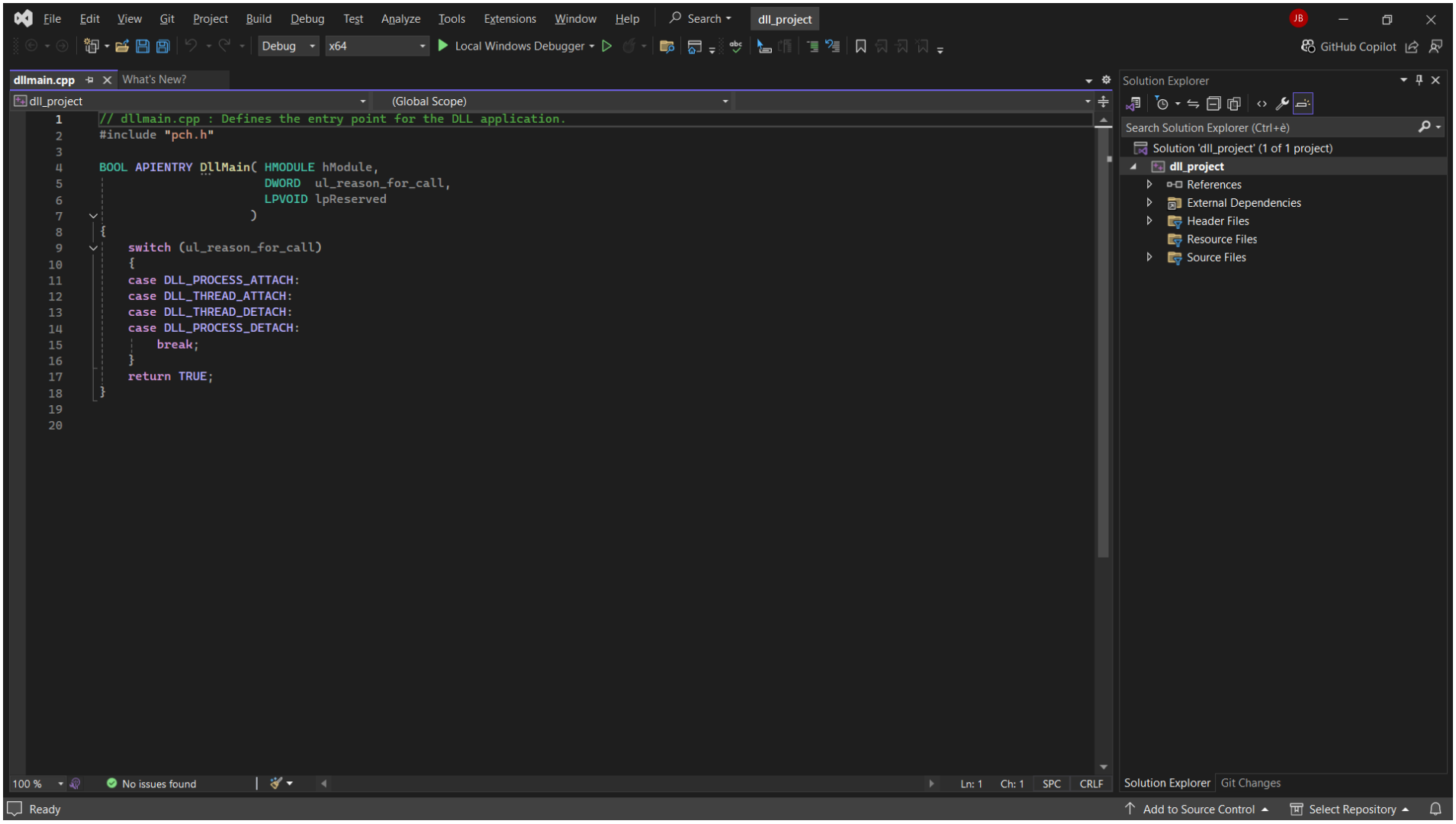
- Search for "dy" and click on first result:



- Click "next" and rename the project as u like, so press the button "create" to finally make our project:



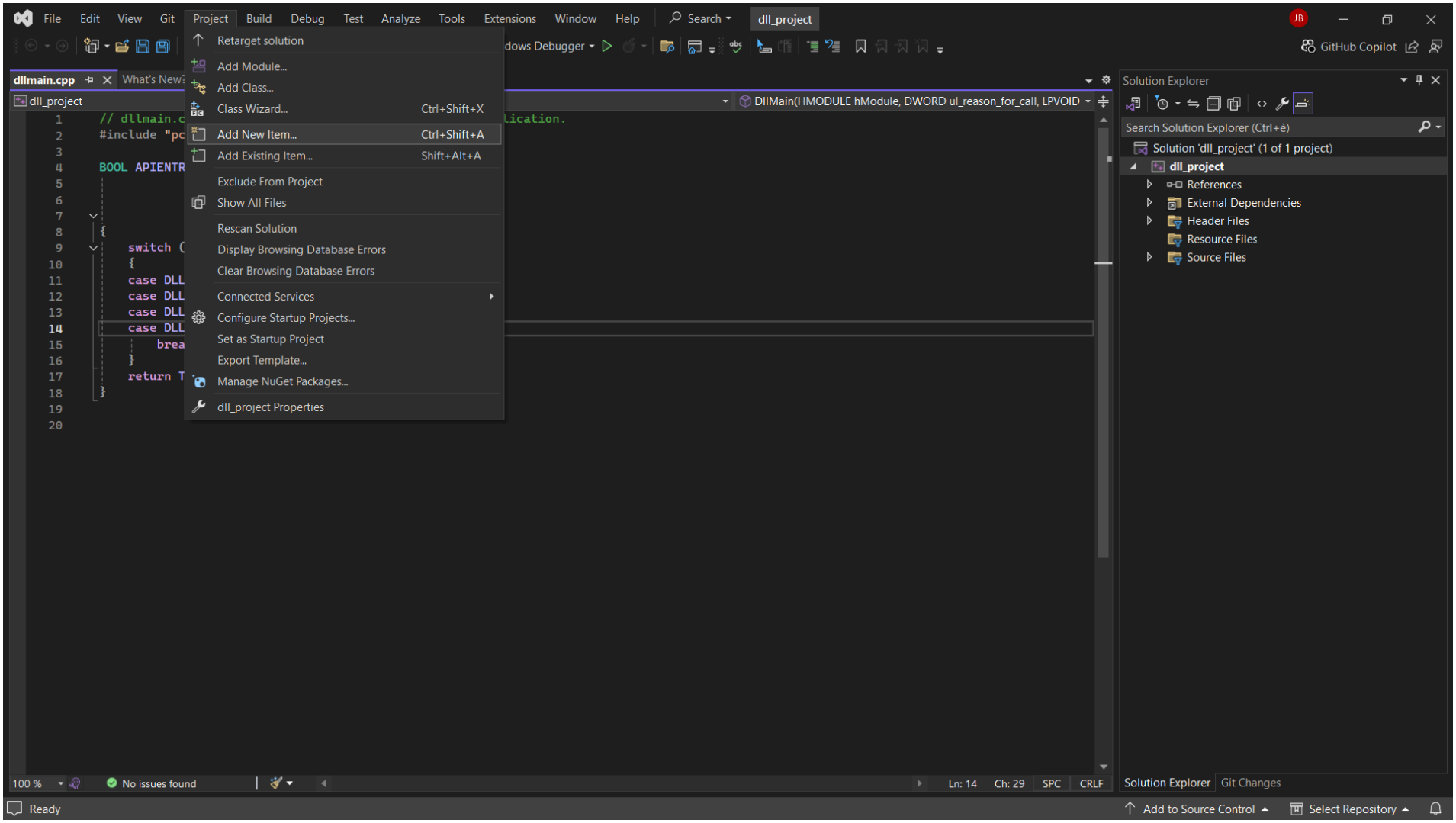
- You should get a similar screen:



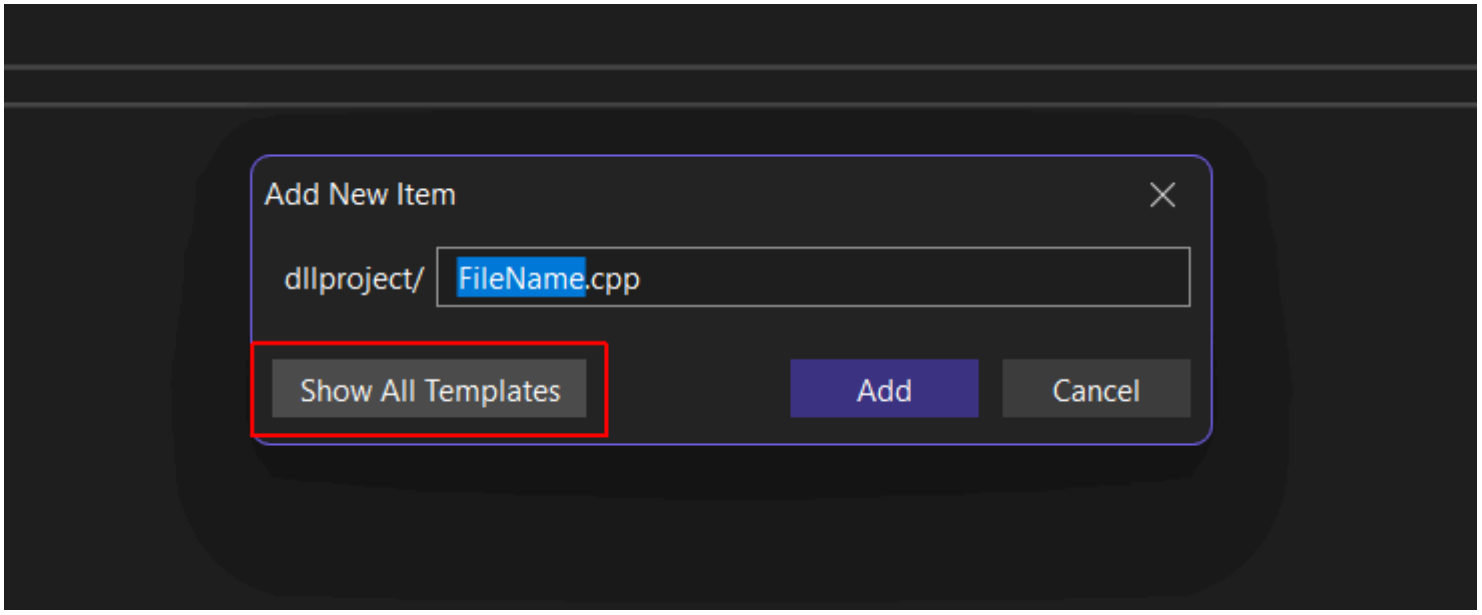
---

5. Create a header file:

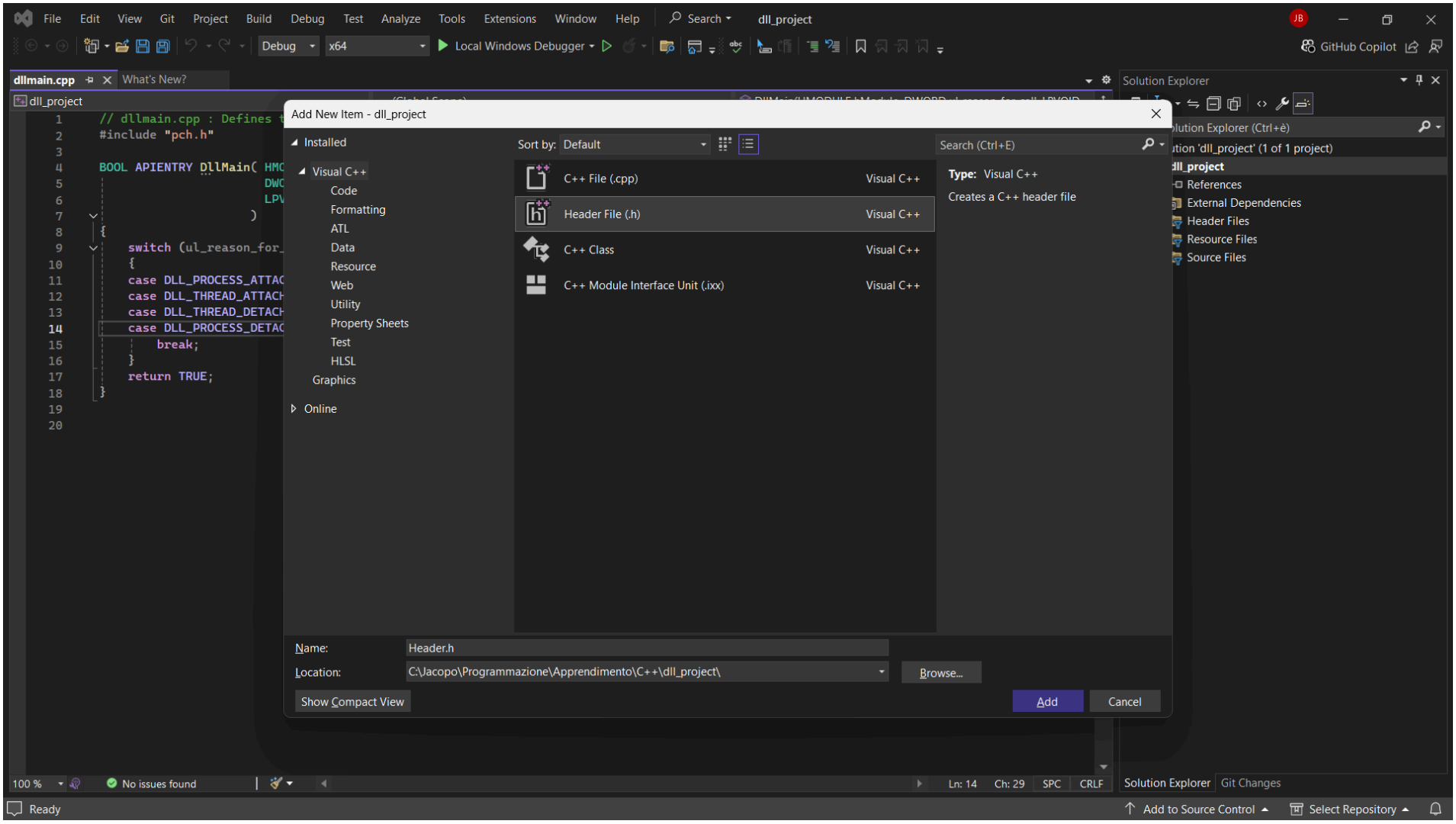
- Press on Project on the menu tab and click to add item:



- On the window that will come up click on "view all templates":



- Choose the Header File and click the button "Add":



- Replace the content with this code:

```
// MathLibrary.h - Contains declarations of math functions
#pragma once

#ifdef MATHLIBRARY_EXPORTS
```

```
#define MATHLIBRARY_API __declspec(dllexport)
#else
#define MATHLIBRARY_API __declspec(dllimport)
#endif

// The Fibonacci recurrence relation describes a sequence F
// where F(n) is { n = 0, a
//               { n = 1, b
//               { n > 1, F(n-2) + F(n-1)
// for some initial integral values a and b.
// If the sequence is initialized F(0) = 1, F(1) = 1,
// then this relation produces the well-known Fibonacci
// sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

// Initialize a Fibonacci relation sequence
// such that F(0) = a, F(1) = b.
// This function must be called before any other function.
extern "C" MATHLIBRARY_API void fibonacci_init(
    const unsigned long long a, const unsigned long long b);

// Produce the next value in the sequence.
// Returns true on success and updates current value and index;
// false on overflow, leaves current value and index unchanged.
extern "C" MATHLIBRARY_API bool fibonacci_next();

// Get the current value in the sequence.
extern "C" MATHLIBRARY_API unsigned long long fibonacci_current();

// Get the position of the current value in the sequence.
extern "C" MATHLIBRARY_API unsigned fibonacci_index();
```

- Explain:

Preprocessor and Macro

pragma once: This command prevents the header file from being included multiple times in a single compilation file, preventing duplication.

ifdef MATHLIBRARY_EXPORTS: This block checks whether MATHLIBRARY_EXPORTS is defined.

define MATHLIBRARY_API __declspec(dllexport): If MATHLIBRARY_EXPORTS is defined (that is, when you are compiling the DLL library), MATHLIBRARY_API is defined as __declspec(dllexport). This specifies that functions declared with this macro should be exported from the DLL, making them available to other applications.
else: If MATHLIBRARY_EXPORTS is not defined (that is, when you are using the DLL library in another application), MATHLIBRARY_API is defined as __declspec(dllimport). This optimizes the import of functions from the DLL.
endif: Closes the ifdef block.
Function Declarations.
extern "C": This command tells the compiler to use C linkage for declared functions, instead of C++, which prevents name decoration (name mangling) and allows compatibility with other languages.

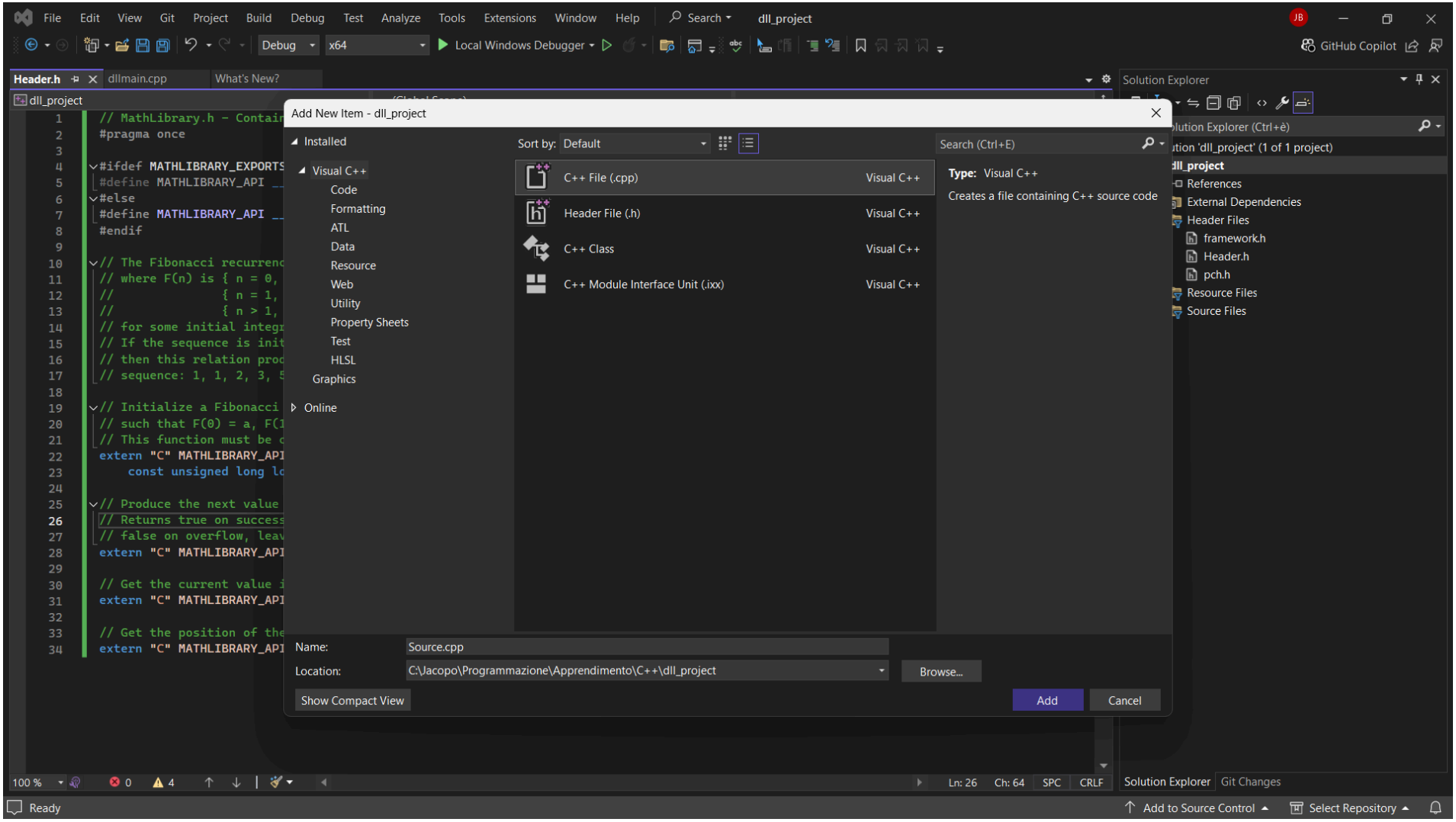
MATHLIBRARY_API: This prefix applies __declspec(dllexport) or __declspec(dllimport) to functions, depending on how the MATHLIBRARY_EXPORTS macro is defined.

Declared functions:

fibonacci_init(const unsigned long long a, const unsigned long b): Initializes the Fibonacci sequence with initial values a and b. Must be called before any other function in the library.
fibonacci_next(): Calculates the next value of the Fibonacci sequence. Returns true if the operation is successful and updates the current value and index. Returns false in case of overflow, leaving the current value and index unchanged.
fibonacci_current(): Returns the current value of the Fibonacci sequence.
fibonacci_index(): Returns the index of the current value in the Fibonacci sequence.
Context of Use
When you compile the DLL library, the MATHLIBRARY_EXPORTS macro is defined in the DLL project, so functions are exported

using __declspec(dllexport). When you include this header file in a project that uses the DLL, MATHLIBRARY_EXPORTS is not defined, so functions are imported using __declspec(dllimport).

- Add a new C++ file in the same way that u add added the header file:



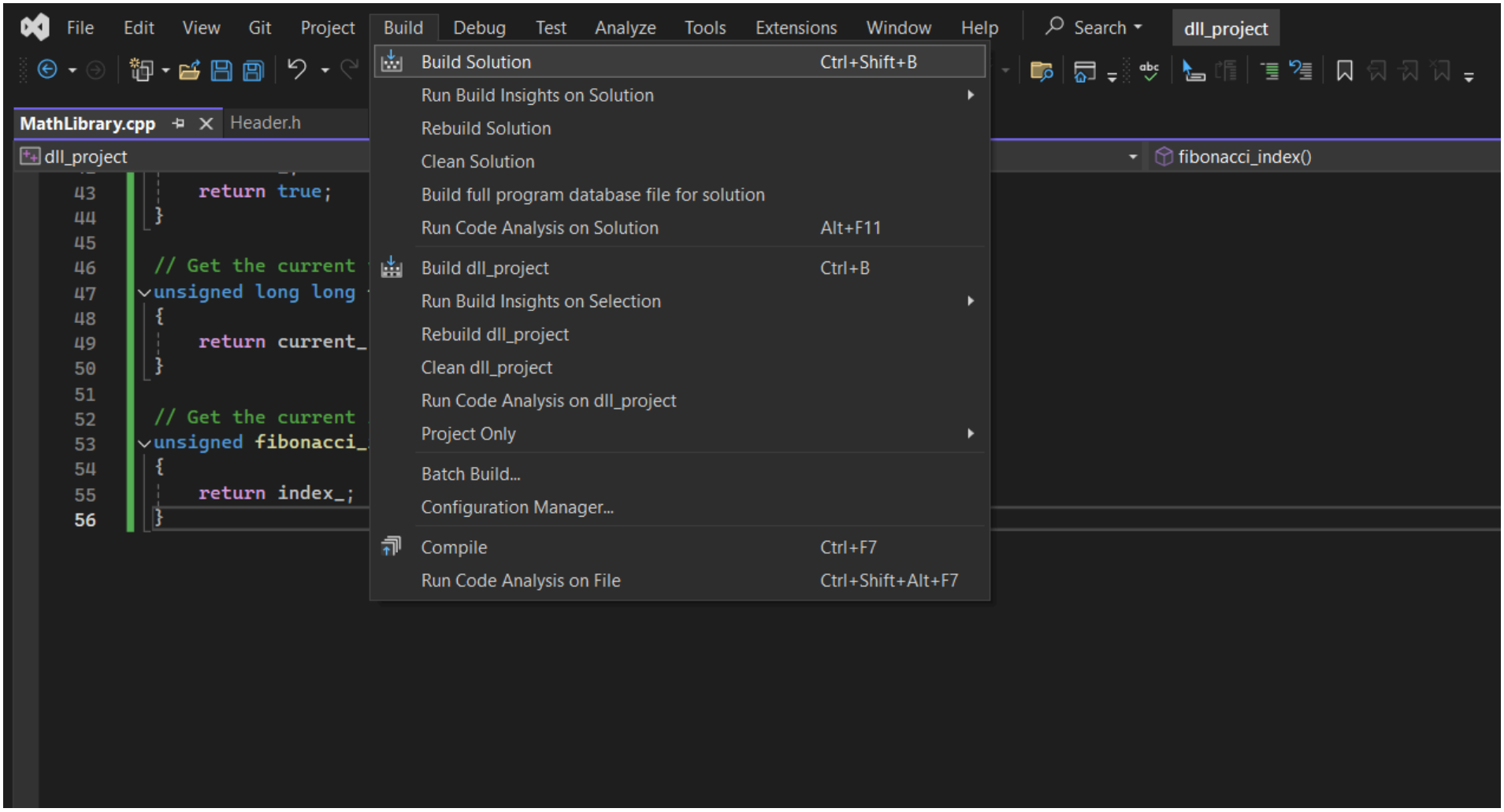
- Copy the following code and paste it into the C++ file:

```
// MathLibrary.cpp : Defines the exported functions for the DLL.
#include "pch.h" // use stdafx.h in Visual Studio 2017 and earlier
```

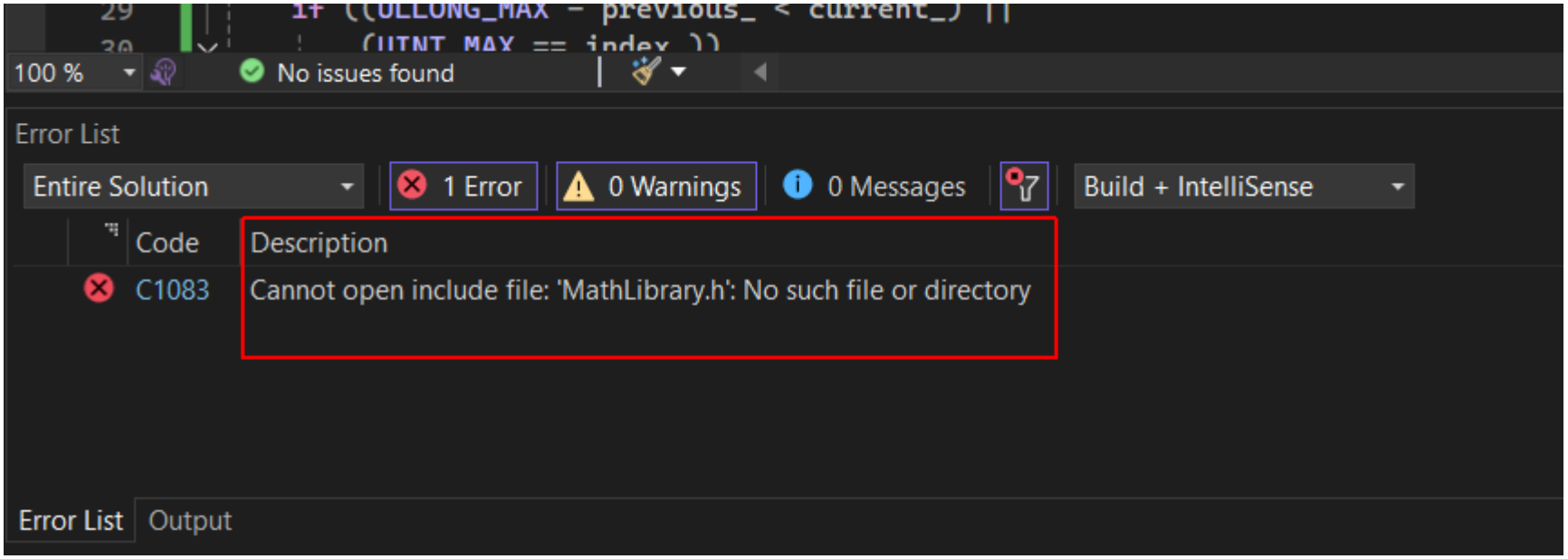
```
#include <utility>
#include <limits.h>
#include "MathLibrary.h"

// DLL internal state variables:
static unsigned long long previous_;  // Previous value, if any
static unsigned long long current_;   // Current sequence value
static unsigned index_;               // Current seq. position

// Initialize a Fibonacci relation sequence
// such that F(0) = a, F(1) = b.
// This function must be called before any other function.
void fibonacci_init(
    const unsigned long long a,
    const unsigned long long b)
{
    index_ = 0;
    current_ = a;
    previous_ = b; // see special case when initialized
}

// Produce the next value in the sequence.
// Returns true on success, false on overflow.
bool fibonacci_next()
{
    // check to see if we'd overflow result or position
    if ((ULLONG_MAX - previous_ < current_) ||
        (UINT_MAX == index_))
    {
        return false;
    }

    // Special case when index == 0, just return b value
    if (index_ > 0)
```

```
    {
        // otherwise, calculate next sequence value
        previous_ += current_;
    }
    std::swap(current_, previous_);
    ++index_;
    return true;
}

// Get the current value in the sequence.
unsigned long long fibonacci_current()
{
    return current_;
}

// Get the current index position in the sequence.
unsigned fibonacci_index()
{
    return index_;
}
```

- Now for testing our code go to the tab **Menu -> Build -> Build solution:



- We will find an error, to fix it we will just rename, in the Menu on the right, our *file_name*.h and rename it "MathLibrary.h":



st Analyze Tools Extensions Window Help Search dll_project

Local Windows Debugger

GitHub Copilot

What's New?

(Global Scope) fibonacci_index()

```
ported functions for the DLL.
 Visual Studio 2017 and earlier



  // Previous value, if any
  // Current sequence value
  // Current seq. position

sequence

ore any other function.



se when initialized


quence.
 overflow.


 result or position
urrent_) ||
```

Ln: 1 Ch: 1 SPC CRLF

Solution Explorer

Search Solution Explorer (Ctrl+è)

- Solution 'dll_project' (1 of 1 project)
  - dll_project
    - References
    - External Dependencies
    - Header Files
      - framework.h
      - MathLibrary.h
      - pch.h
    - Resource Files
    - Source Files
      - dllmain.cpp
      - MathLibrary.cpp
      - pch.cpp

) Messages Build + IntelliSense

Search Error List

	Project	File	Line	Details
file or directory	dll_project	MathLibrary.cpp	5	

- Now retry to Build our C++ file; an output like this should appear in the console:

```
Output
Show output from: Build
1>C:\Jacopo\Programmazione\Apprendimento\C++\dll_project\MathLibrary.cpp(53,10): wa
1>   Creating library C:\Jacopo\Programmazione\Apprendimento\C++\dll_project\x64\De
1>dll_project.vcxproj -> C:\Jacopo\Programmazione\Apprendimento\C++\dll_project\x64
1>Done building project "dll_project.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 15:20 and took 01,074 seconds ==========
```

---

6. Create a client app that uses our DLL:

- Explain:

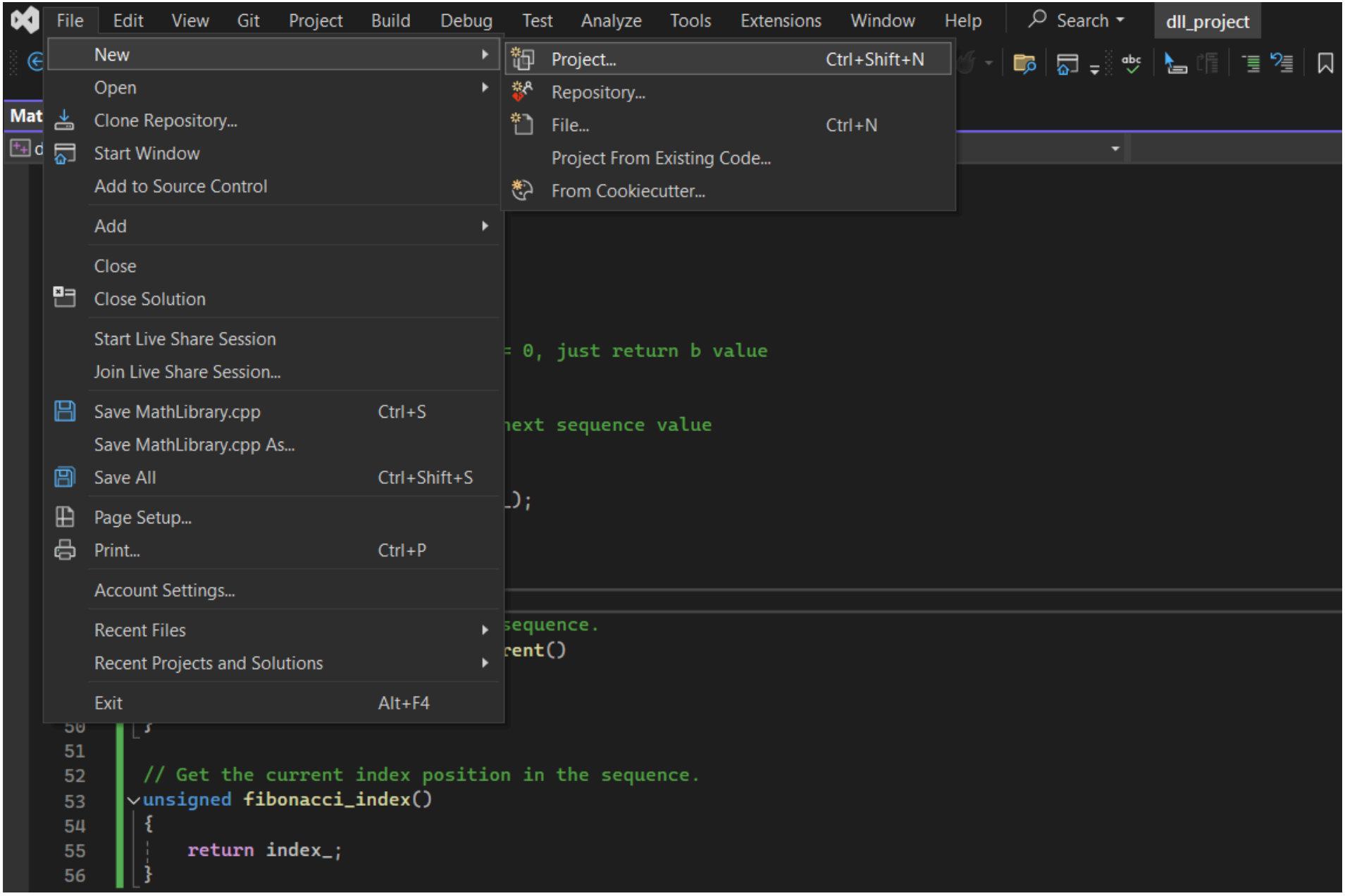
  When you create a DLL, think about how client apps may use it. To call the functions or access the data exported by a DLL, client source code must have the declarations available at compile time. At link time, the linker requires information to resolve the function calls or data accesses. A DLL supplies this information in an *import library*, a file that contains information about how to find the functions and data, instead of the actual code. And at run time, the DLL must be available to the client, in a location that the operating system can find.

  Whether it's your own or from a third-party, your client app project needs several pieces of information to use a DLL. It needs to find the headers that declare the DLL exports, the import libraries for the linker, and the DLL itself. One solution is to copy all of these files into your client project. For third-party DLLs that are unlikely to change while your client is in development, this method may be the best way to use them. However, when you also build the DLL, it's better to avoid duplication. If you make a local copy of DLL files that are under development, you may accidentally change a header file in one copy but not the other, or use an out-of-date library.

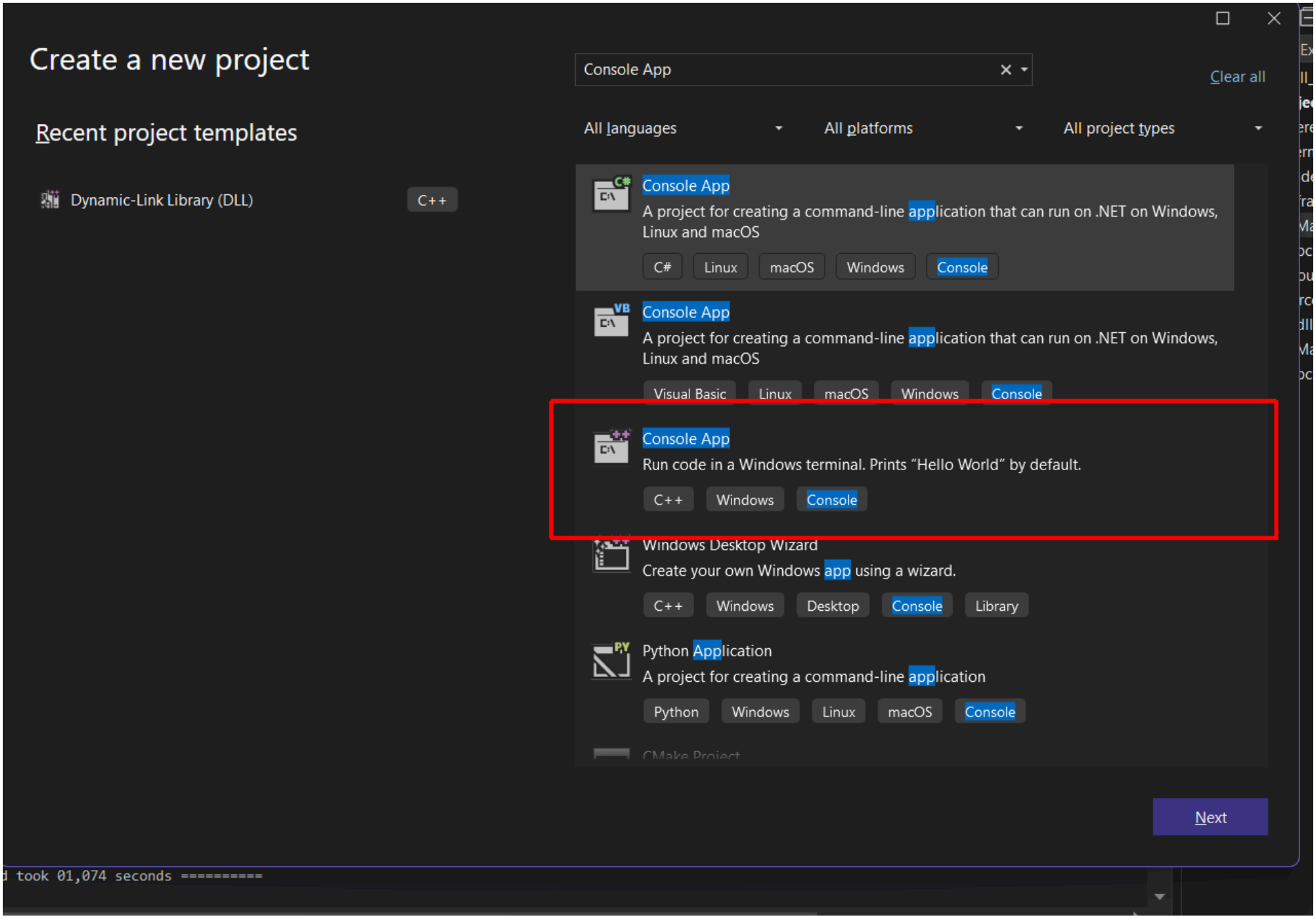
  To avoid out-of-sync code, we recommend you set the include path in your client project to include the DLL header files directly from your DLL project. Also, set the library path in your client project to include the DLL import libraries from the DLL project. And finally,

copy the built DLL from the DLL project into your client build output directory. This step allows your client app to use the same DLL code you build.

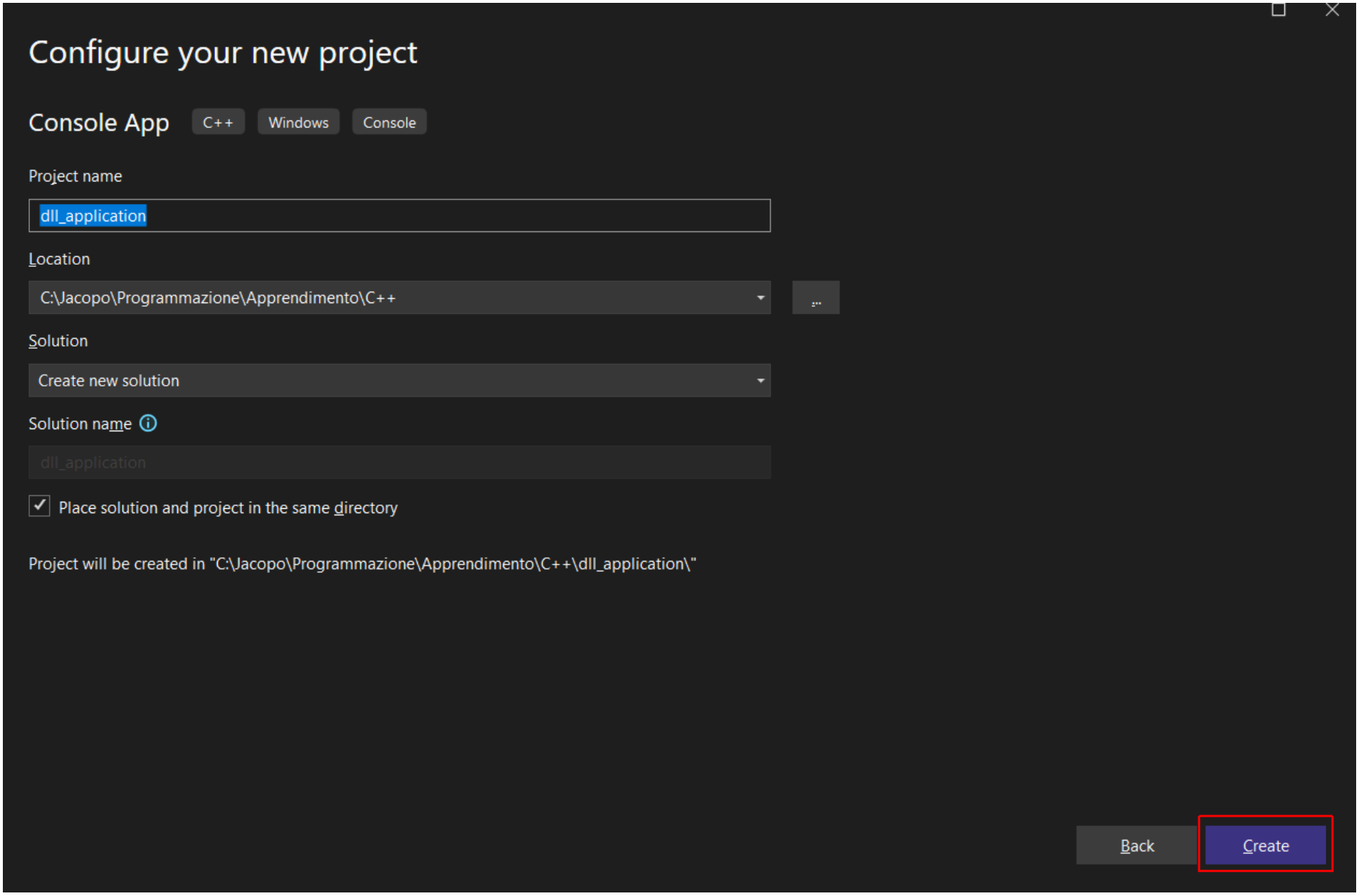
- On the Menu Bar choose **File** -> **New** -> **Project** and create a new project:



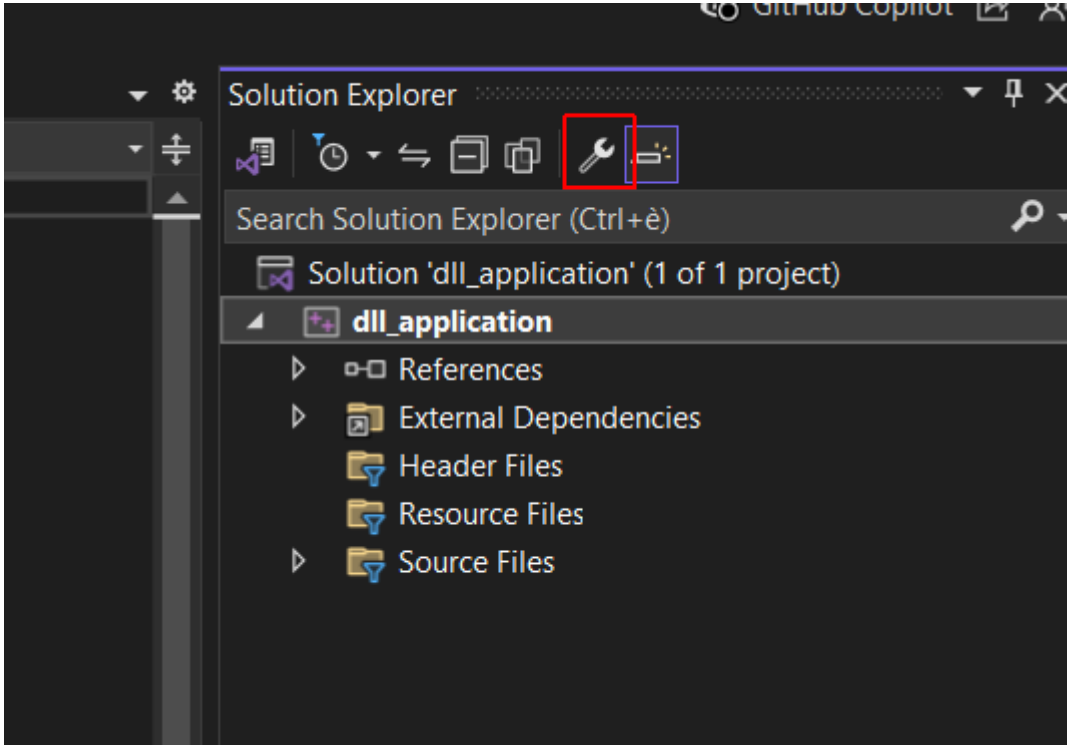
- Search for "Console App" in C++:



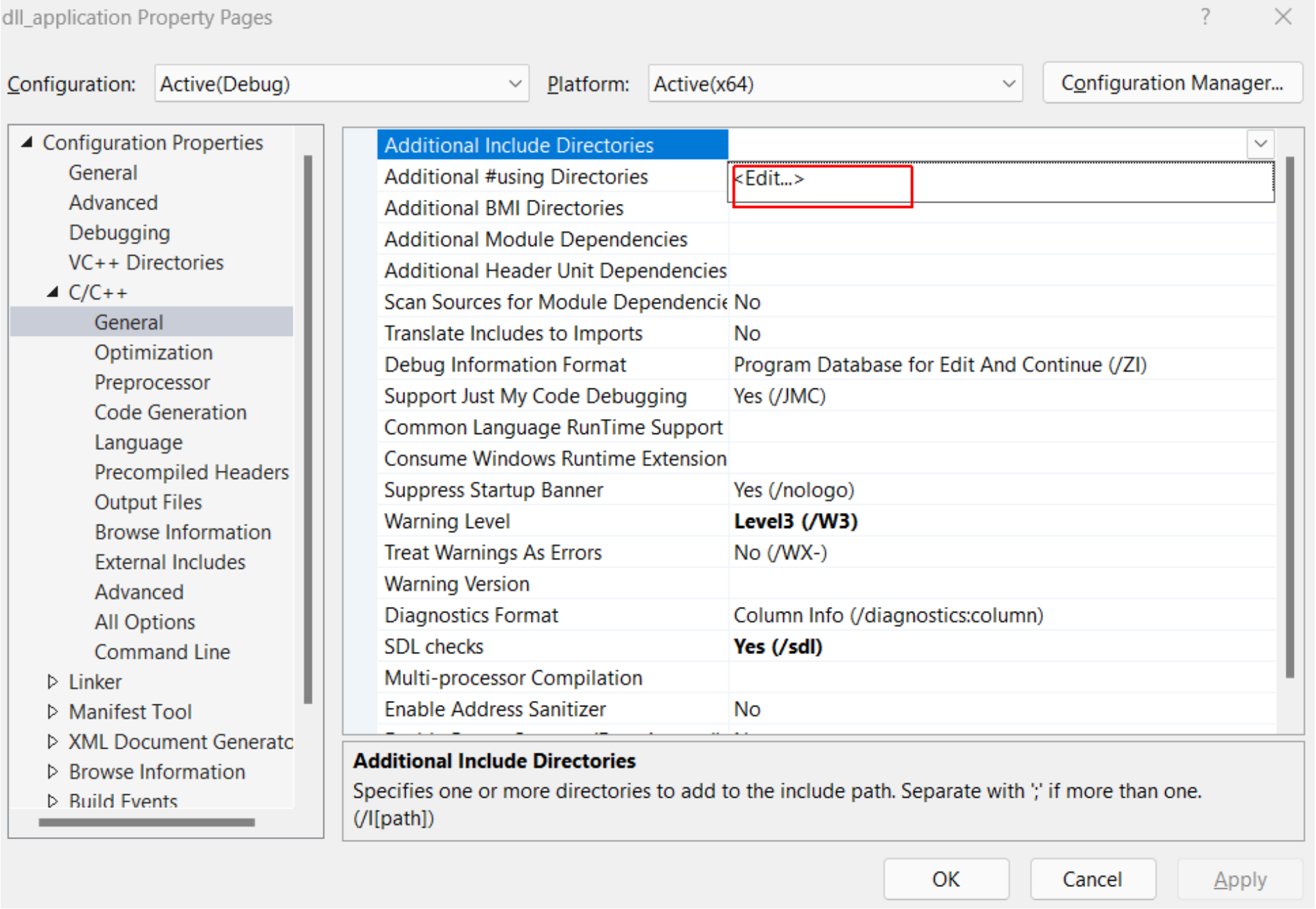
- Rename it and choose the Create button:



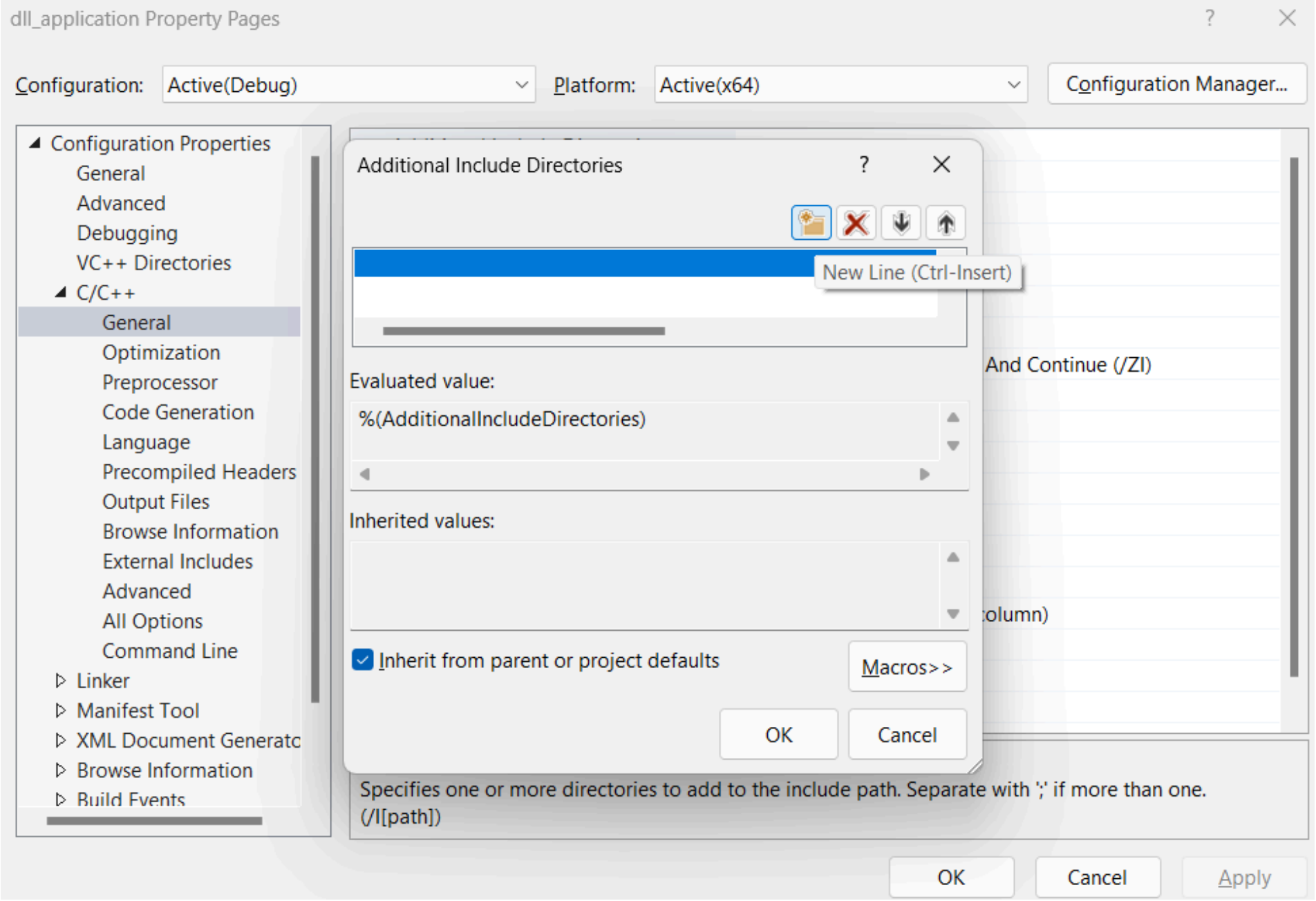
- Add DLL to our application; Go to **Solution Explorer -> Properties**:


GitHub Copilot
Solution Explorer
Search Solution Explorer (Ctrl+è)
Solution 'dll_application' (1 of 1 project)
dll_application
References
External Dependencies
Header Files
Resource Files
Source Files

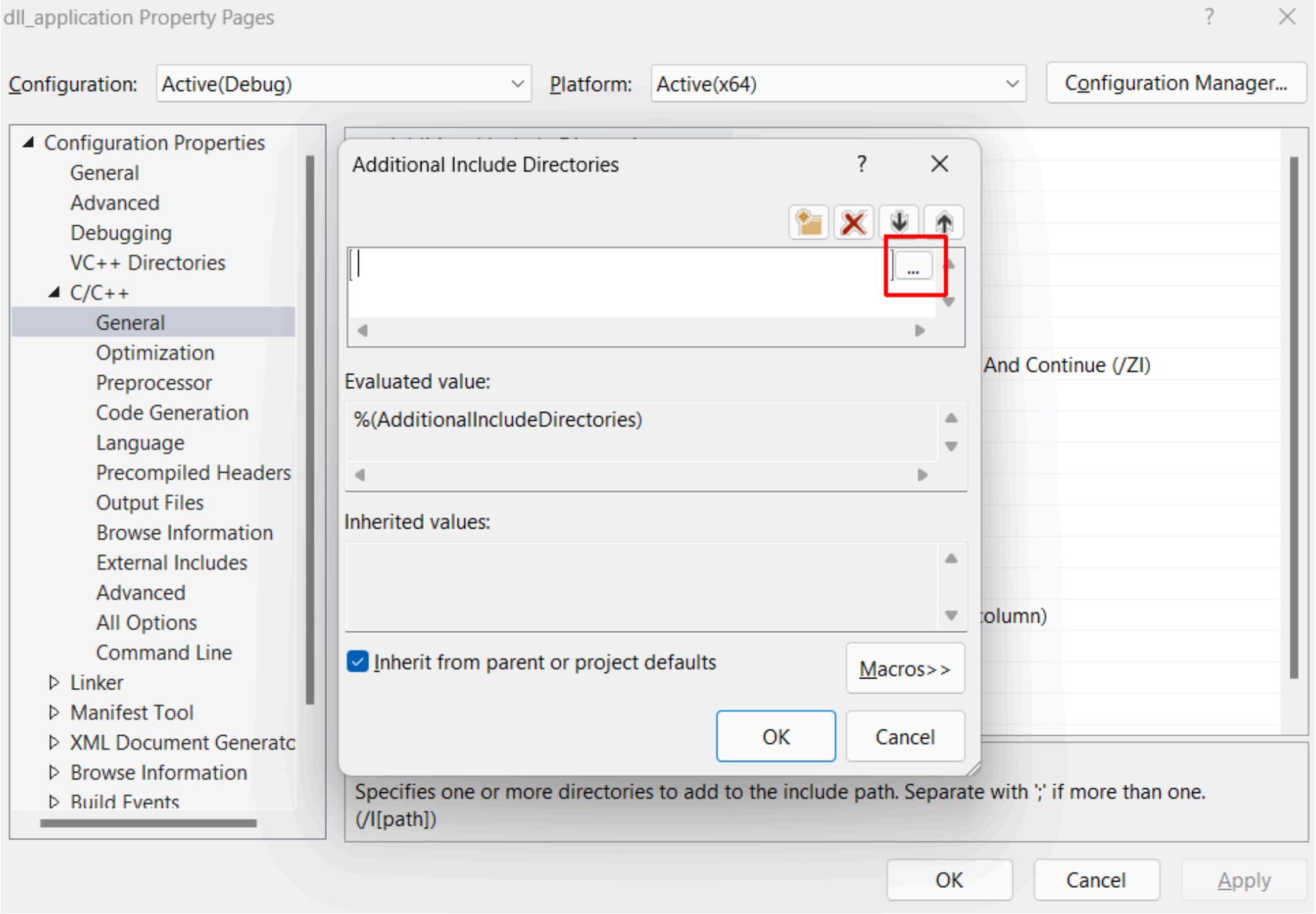
- Go to **C/C++** -> **General** -> **Additional Include Directories** and choose the option **Edit**:



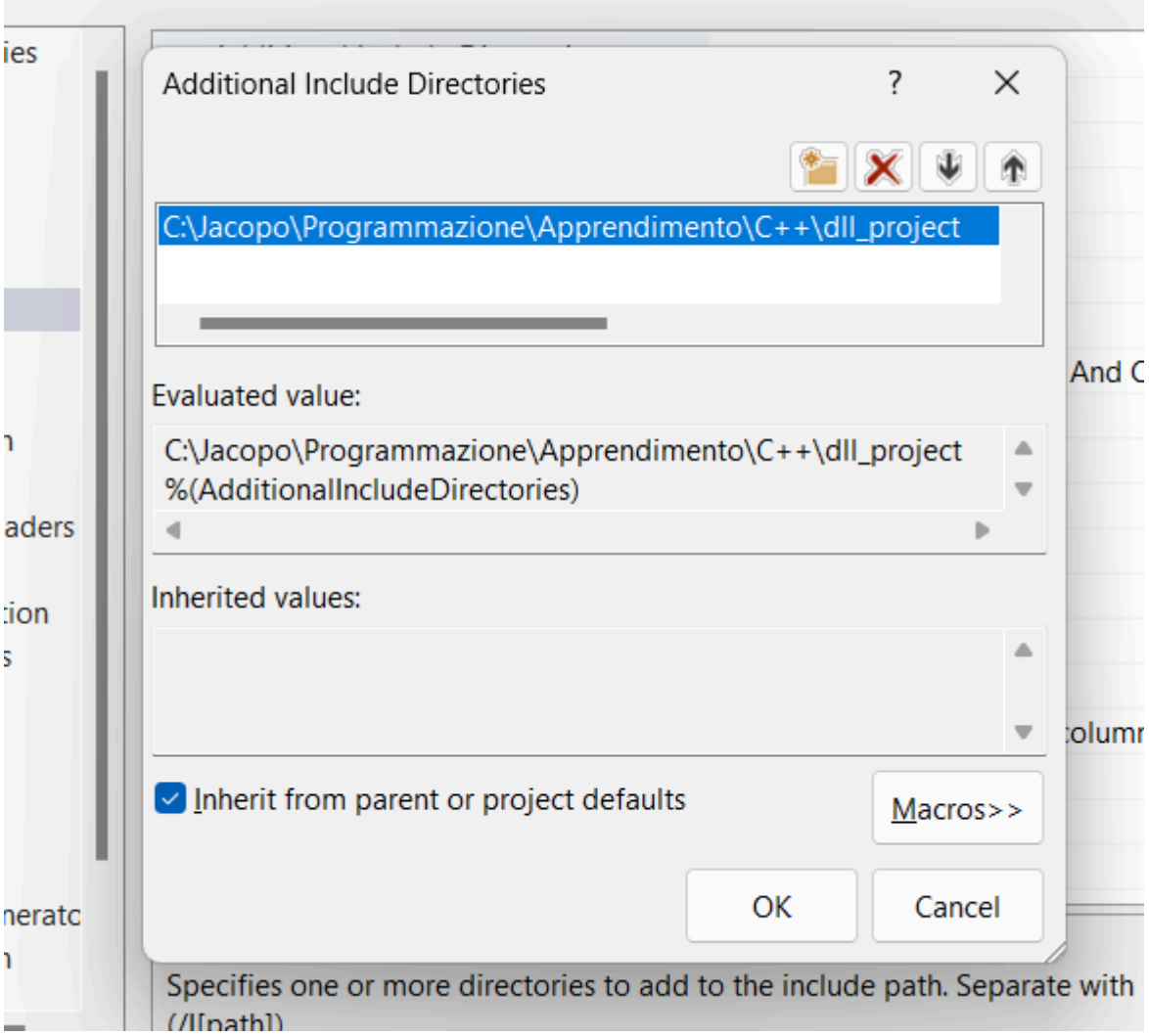
- Select the folder icon:



- Go over the line that u created and select (at the right) the three dots:



- Now select the path to your DLL project:



- At finish press **Ok** -> **Apply** -> **Ok**
- In your application code paste this code when is included our **MathLibrary.h** file:

```
// MathClient.cpp : Client app for MathLibrary DLL.
// #include "pch.h" Uncomment for Visual Studio 2017 and earlier
#include <iostream>
```

```
#include "MathLibrary.h"

int main()
{
        // Initialize a Fibonacci relation sequence.
        fibonacci_init(1, 1);
        // Write out the sequence values until overflow.
        do {
                std::cout << fibonacci_index() << ": "
                        << fibonacci_current() << std::endl;
        } while (fibonacci_next());
        // Report count of values written before overflow.
        std::cout << fibonacci_index() + 1 <<
                " Fibonacci sequence values fit in an " <<
                "unsigned 64-bit integer." << std::endl;
}
```

- Info:

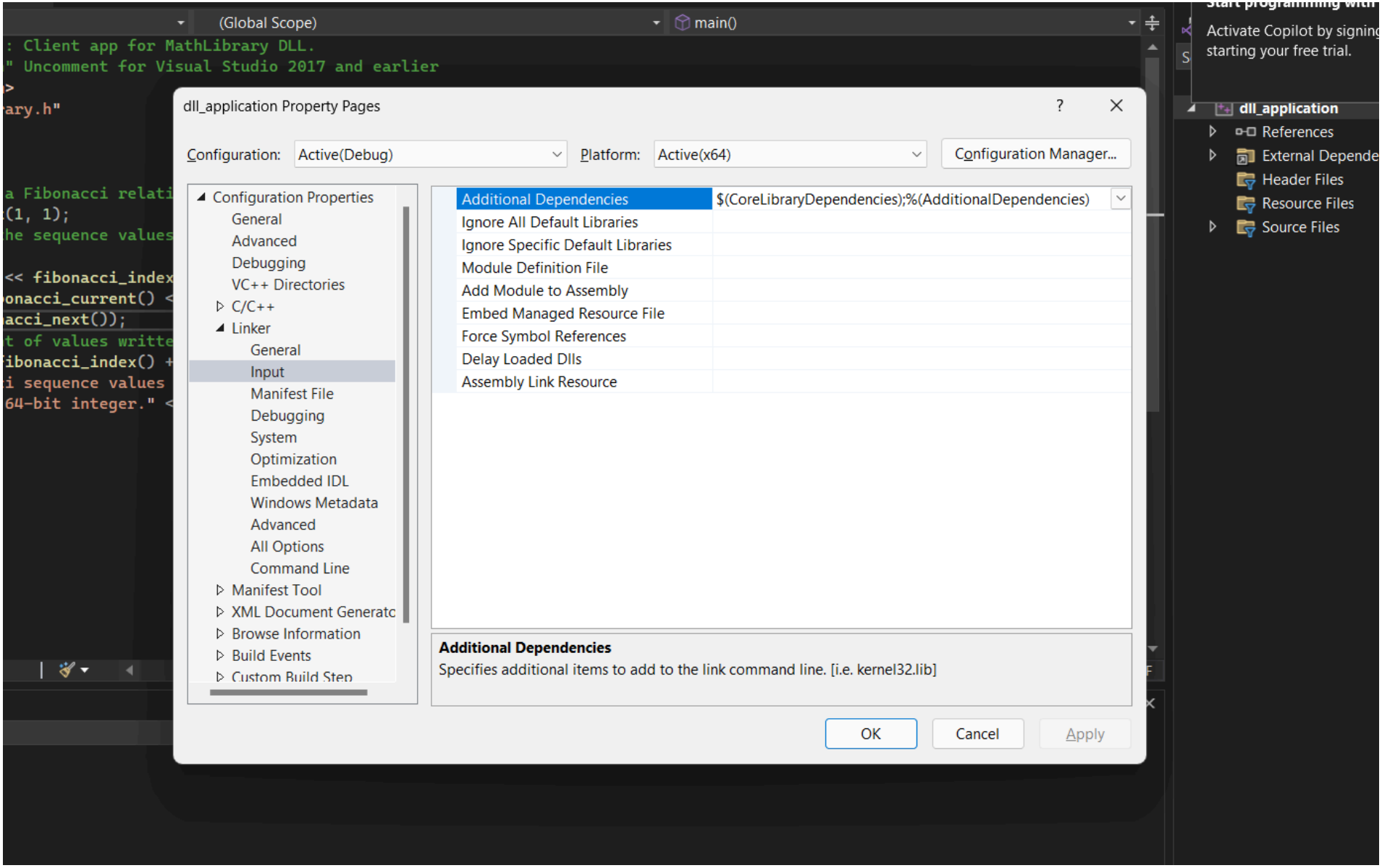
  This code can be compiled, but not linked. If you build the client app now, the error list shows several LNK2019 errors. That's because your project is missing some information: You haven't specified that your project has a dependency on the *MathLibrary.lib* library yet. And, you haven't told the linker how to find the *MathLibrary.lib* file.

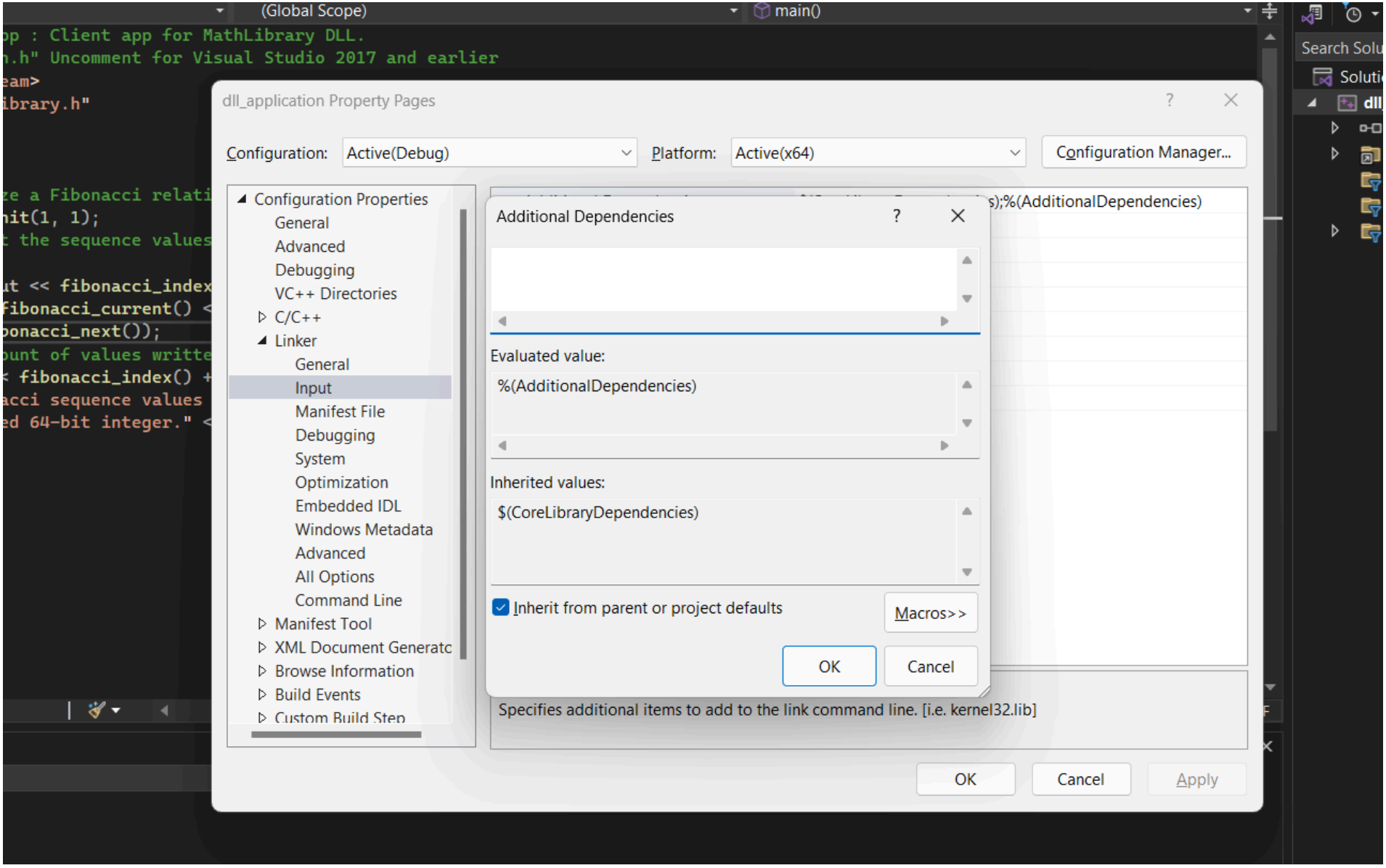
  To fix this issue, you could copy the library file directly into your client app project. The linker would find and use it automatically. However, if both the library and the client app are under development, that might lead to changes in one copy that aren't shown in the other. To avoid this issue, you can set the **Additional Dependencies** property to tell the build system that your project depends on *MathLibrary.lib*. And, you can set an **Additional Library Directories** path in your project to include the path to the original library when you link.

---

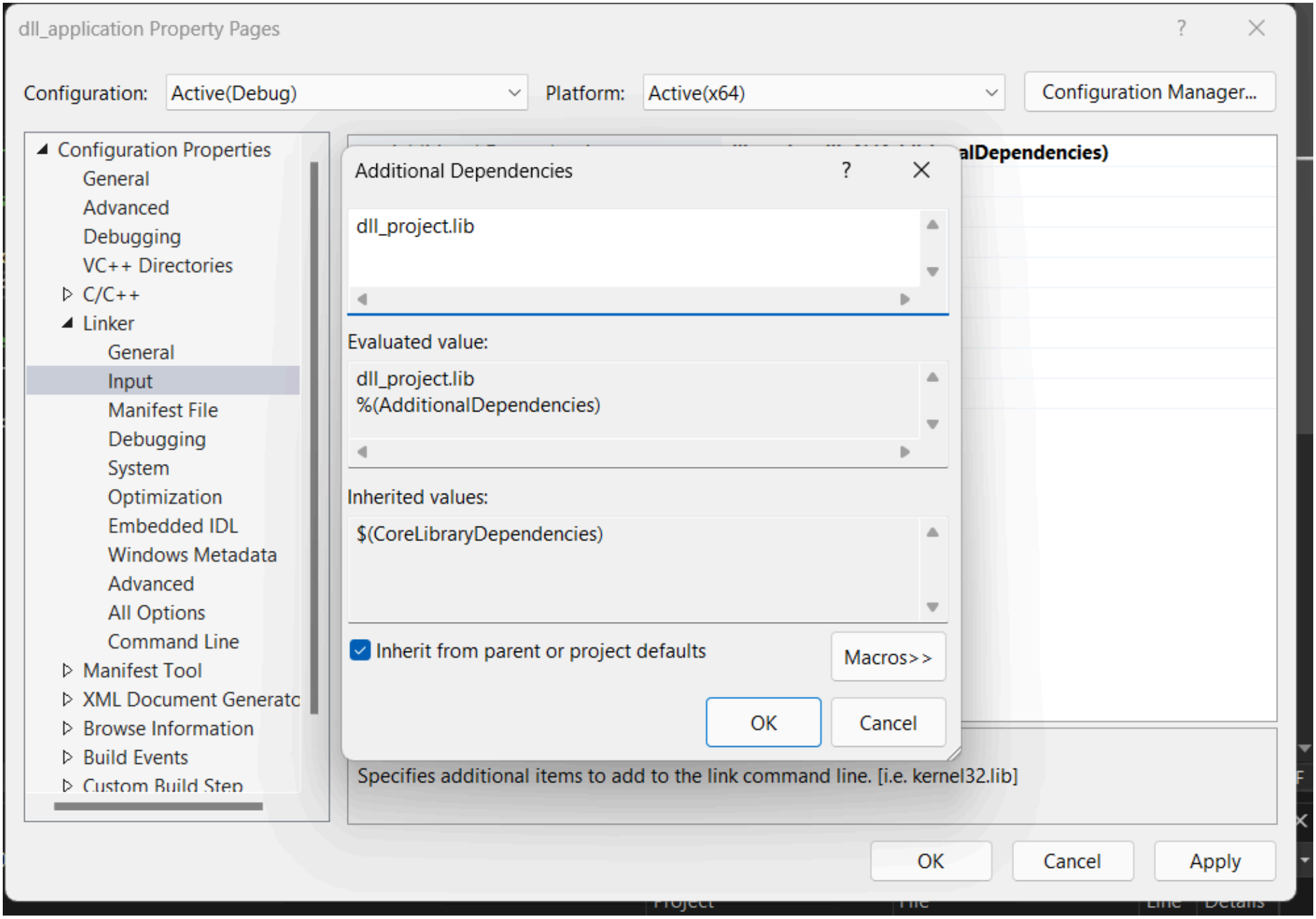
7. Import DLL in our application:

- Go to **Property** ^81bae6 -> **Linker** -> **Input** -> **Additional Dependencies** and choose the option **Edit**:

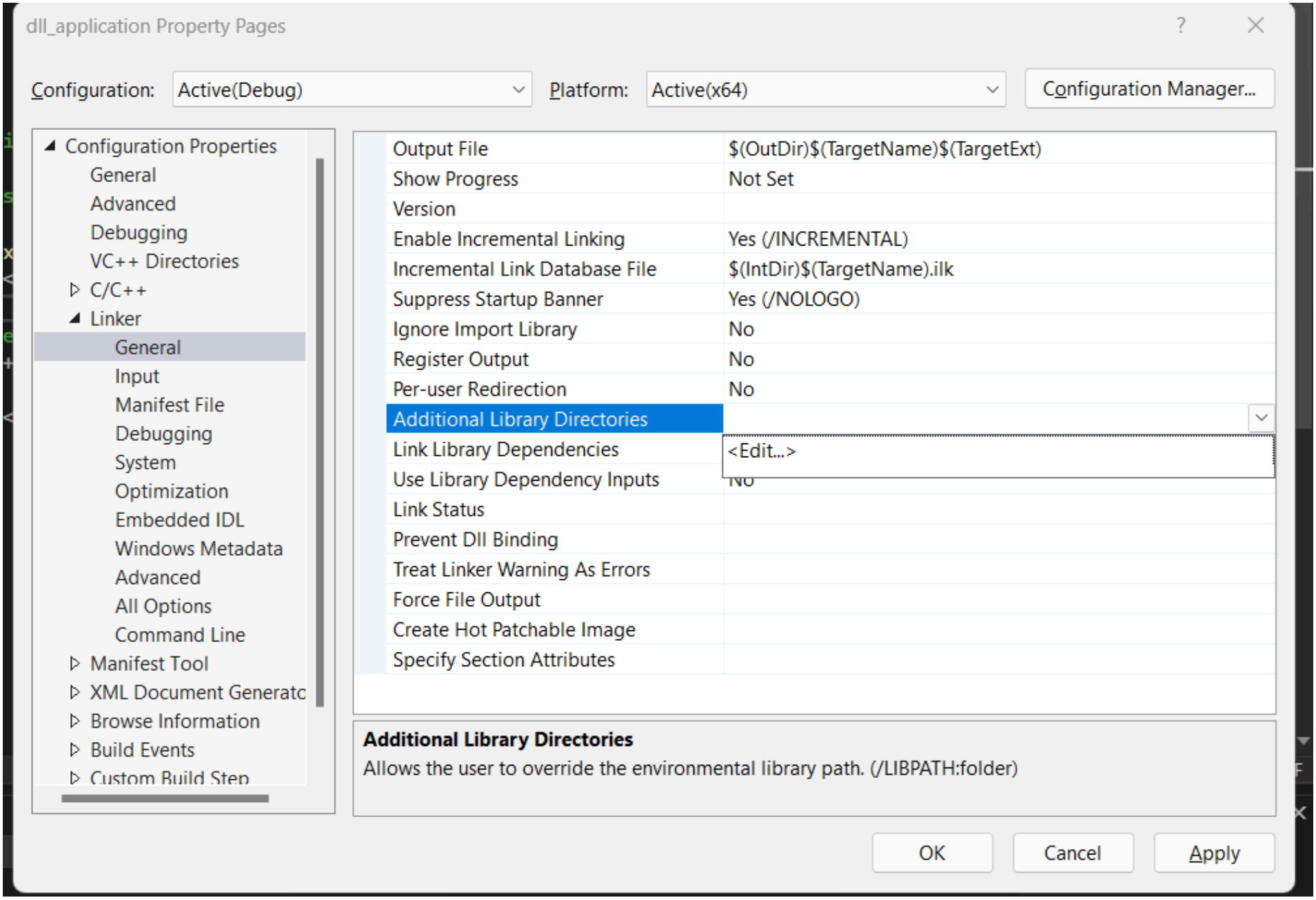

(Global Scope)
main()
: Client app for MathLibrary DLL.
" Uncomment for Visual Studio 2017 and earlier
ary.h"
a Fibonacci relati
(1, 1);
he sequence values
<< fibonacci_index
onacci_current() <
acci_next());
t of values writte
ibonacci_index() +
i sequence values
64-bit integer." <
Activate Copilot by signing
starting your free trial.
dll_application
References
External Depende
Header Files
Resource Files
Source Files
dll_application Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generatc
Browse Information
Build Events
Custom Build Step
Additional Dependencies
$(CoreLibraryDependencies);%(AdditionalDependencies)
Ignore All Default Libraries
Ignore Specific Default Libraries
Module Definition File
Add Module to Assembly
Embed Managed Resource File
Force Symbol References
Delay Loaded Dlls
Assembly Link Resource
Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]
OK
Cancel
Apply


(Global Scope)
main()
dll_application Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generatc
Browse Information
Build Events
Custom Build Step
Additional Dependencies
Evaluated value:
%(AdditionalDependencies)
Inherited values:
$(CoreLibraryDependencies)
Inherit from parent or project defaults
Macros>>
OK
Cancel
Specifies additional items to add to the link command line. [i.e. kernel32.lib]
OK
Cancel
Apply

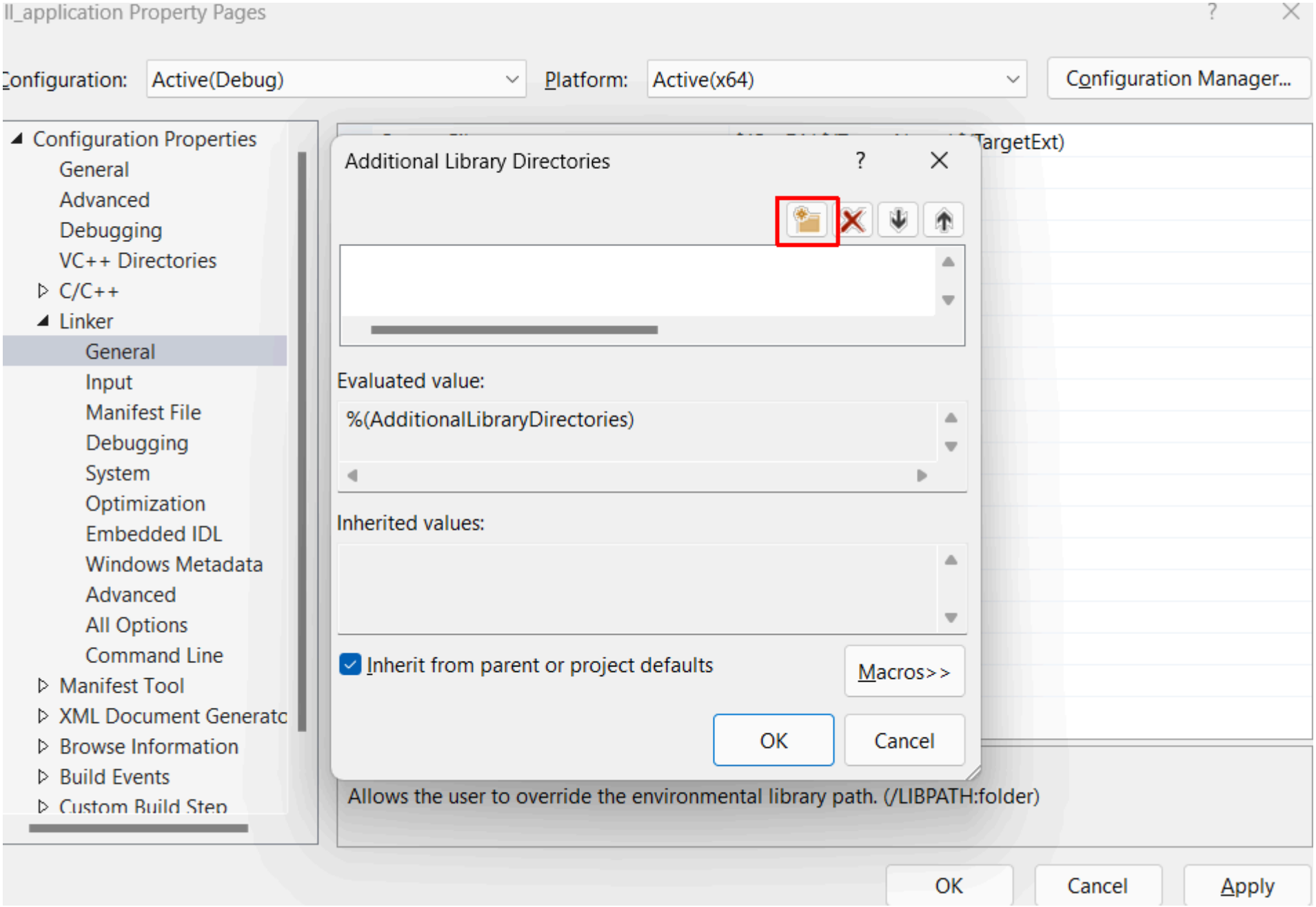
- Into the Additional Dependencies writes **YourProjectName.lib** and click **Ok**:

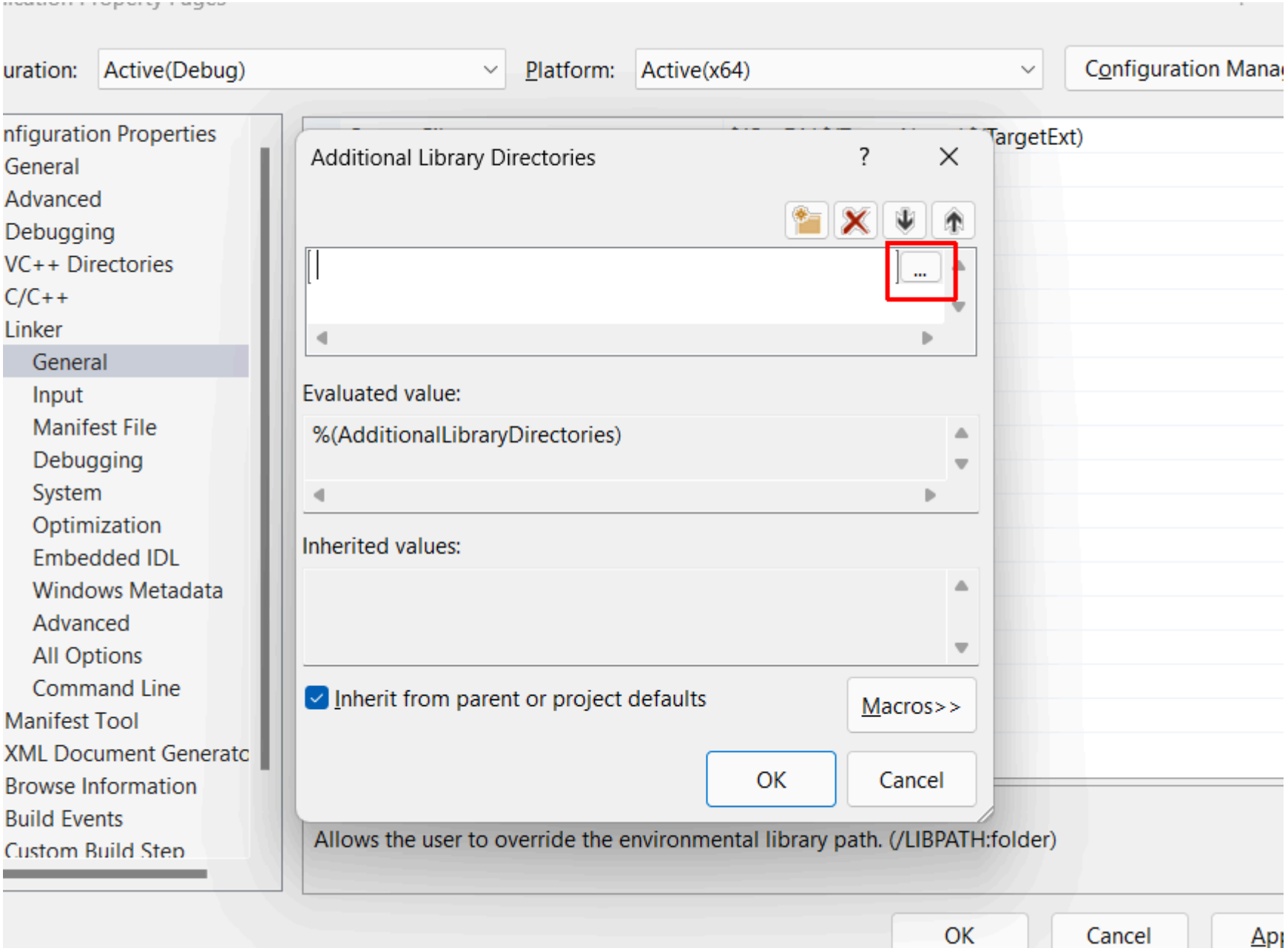


- Now go to **General -> Additional Library Directories** and choose the option **Edit**:



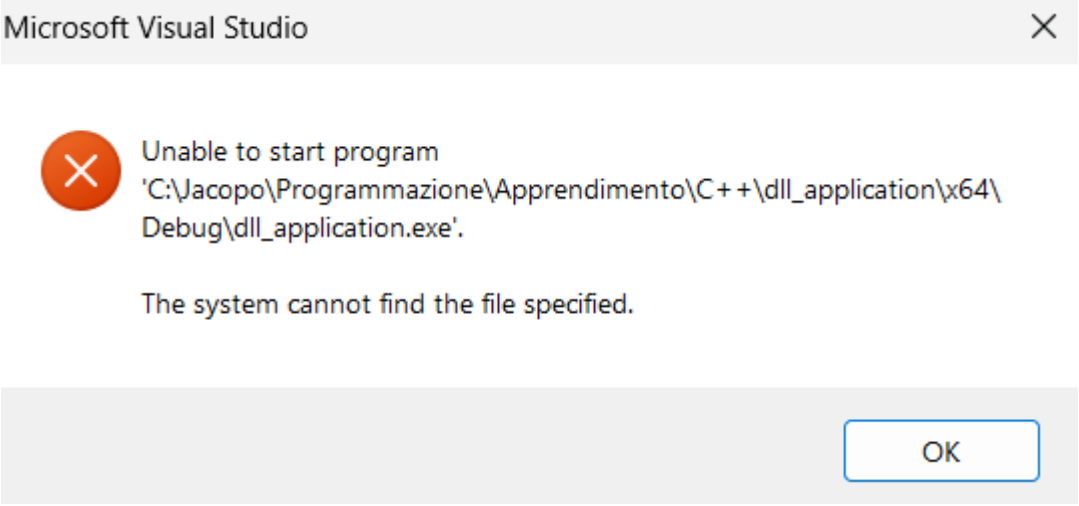
- In this window select the **folder icon** and after click on the **three dots**:





- Select your DLL directory files, The path to our DLL project should look something like this: `....\dll_project\x64\Debug

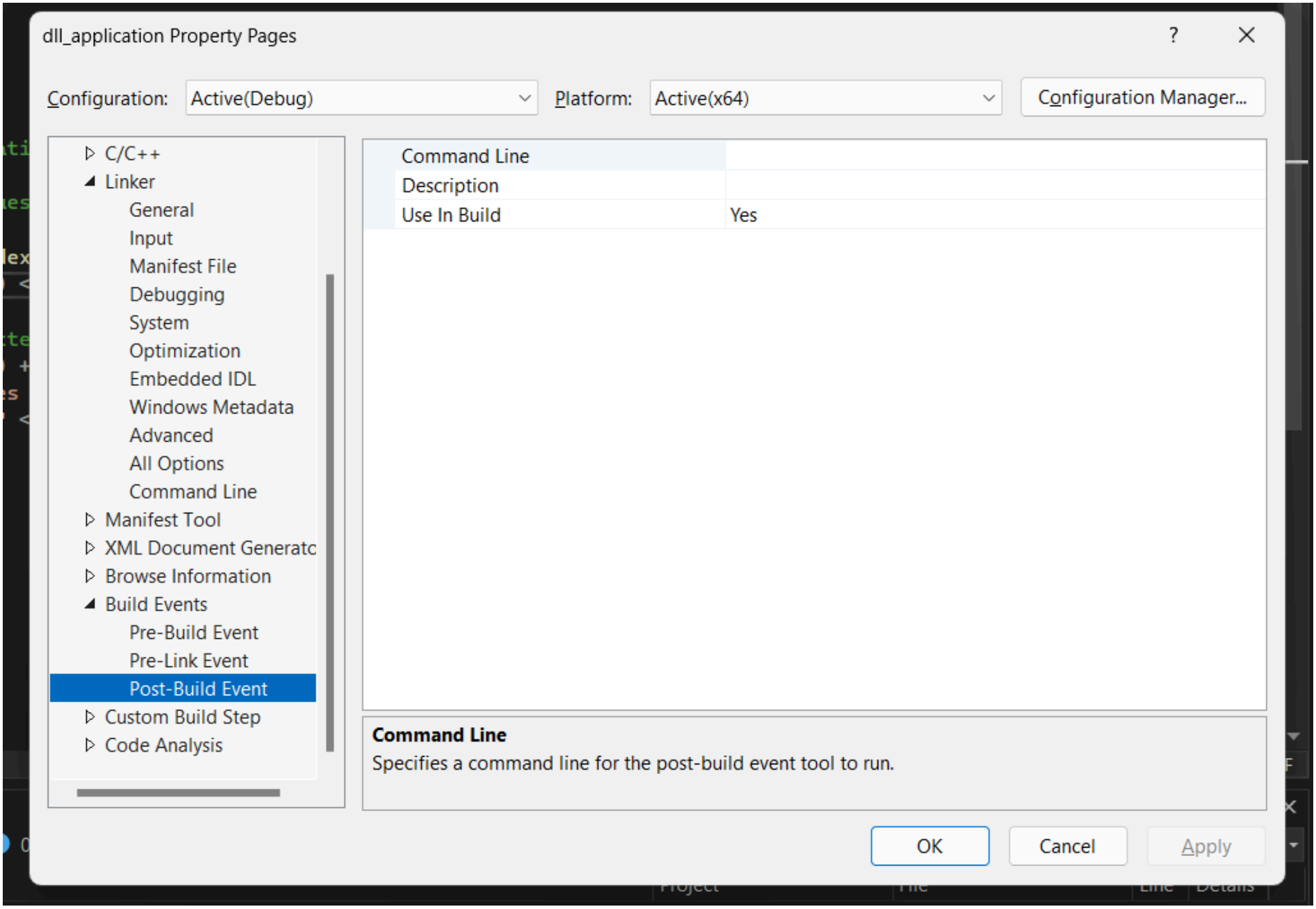
- You will probably get an error like this:



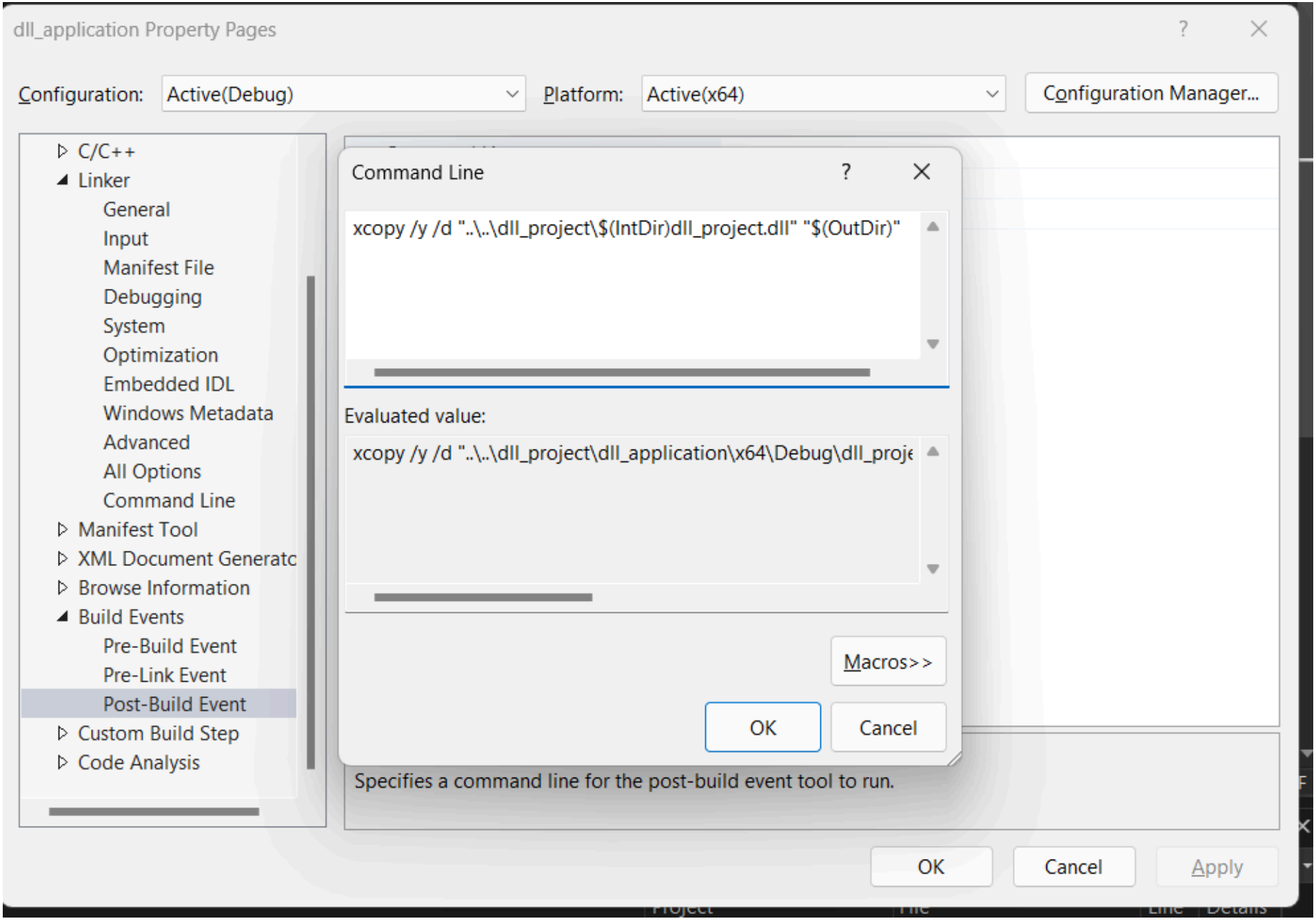
---

8. Last steps to copy our DLL:

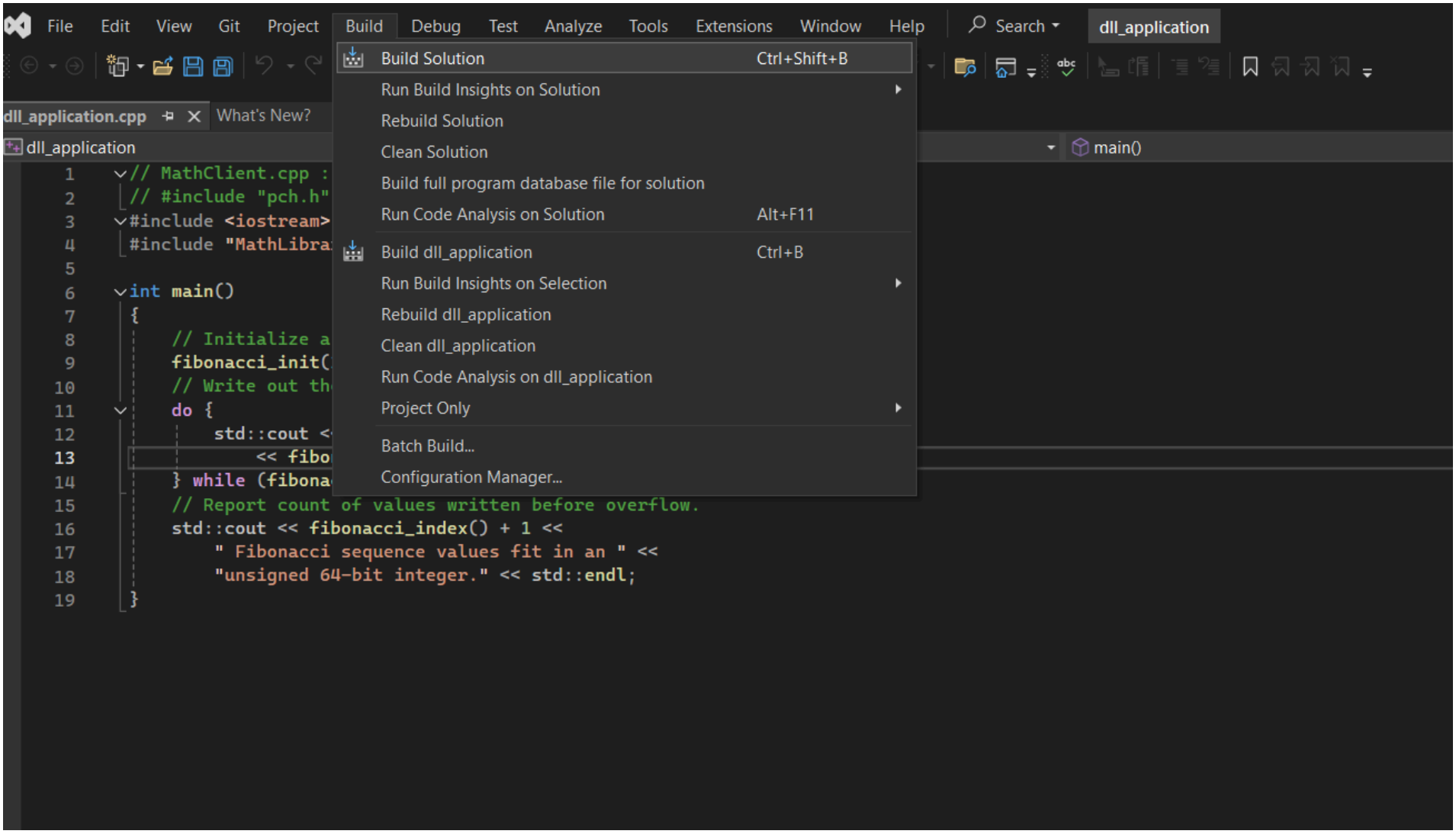
- Go to **Property** -> **Built Event** -> **Post-Built Event** -> **Command Line** and choose the option **Edit**:



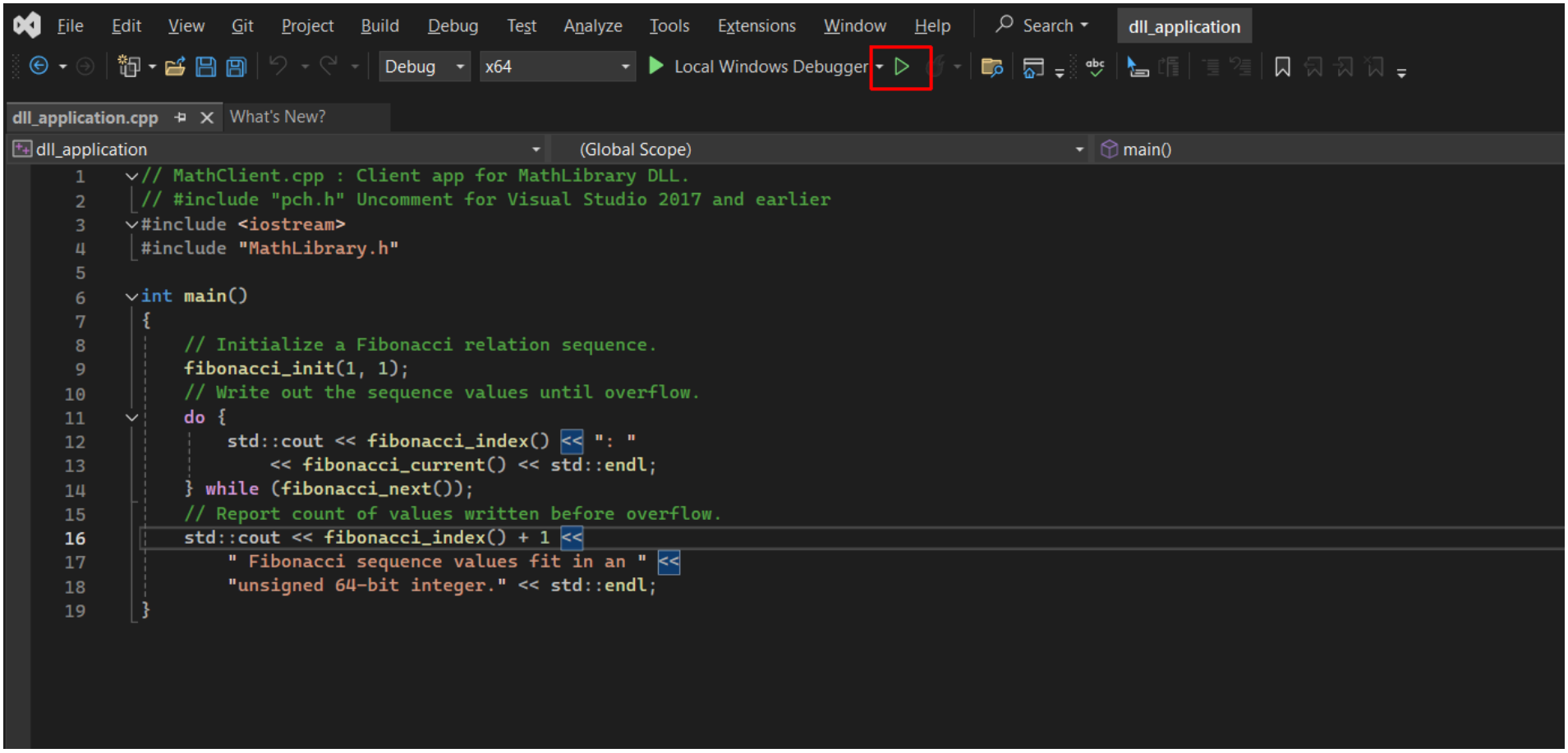
- Into the pane copy the code below and click **Ok** -> **Apply** -> **Ok**:
  `xcopy /y /d "..\..\dll_project\x64\Debug"`


dll_application Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generatc
Browse Information
Build Events
Pre-Build Event
Pre-Link Event
Post-Build Event
Custom Build Step
Code Analysis
Command Line
xcopy /y /d "..\..\dll_project\$(IntDir)dll_project.dll" "$(OutDir)"
Evaluated value:
xcopy /y /d "..\..\dll_project\dll_application\x64\Debug\dll_proje
Macros>>
OK
Cancel
Specifies a command line for the post-build event tool to run.
OK
Cancel
Apply

- Finally on the **Menu bar** click **Build** -> **Build Solution**:



- Now start the program on the triangle icon:



---

9. End:

- Once you run the program you should get a result like this:

```
68: 117669030460994
69: 190392490709135
70: 308061521170129
71: 498454011879264
72: 806515533049393
73: 1304969544928657
74: 2111485077978050
75: 3416454622906707
76: 5527939700884757
77: 8944394323791464
78: 14472334024676221
79: 23416728348467685
80: 37889062373143906
81: 61305790721611591
82: 99194853094755497
83: 160500643816367088
84: 259695496911122585
85: 420196140727489673
86: 679891637638612258
87: 1100087778366101931
88: 1779979416004714189
89: 2880067194370816120
90: 4660046610375530309
91: 7540113804746346429
92: 12200160415121876738
93 Fibonacci sequence values fit in an unsigned 64-bit integer.

C:\Jacopo\Programmazione\Apprendimento\C++\dll_application\x64\Debug\dll_application.exe (process 15916) exited with cod
e 0.
Press any key to close this window . . .
```