

Unit tests for C/C++ in Visual Studio (Microsoft Unit Testing Framework for C/C++)

can write and run your C++ unit tests by using the Test Explorer window.

Some features such as Live Unit Testing, Coded UI Tests and IntelliTest are not supported for C++.

Visual Studio includes these C++ test frameworks with no additional downloads required:

- Microsoft Unit Testing Framework for C++
- Google Test
- Boost.Test
- CTest

Along with using the installed frameworks, you can write your own test adapter for whatever framework you would like to use within Visual Studio.

A test adapter can integrate unit tests with the Test Explorer window.

Basic test workflow

The following sections show the basic steps to get you started with C++ unit testing.

Create a test project in Visual Studio

Can define and run tests inside one or more test projects.

Create the projects in the same solution as the code you want to test.

Create a console type project in visual studio. Ex: MyProgram

The first screenshot shows the `Math.h` header file. It contains a `#pragma once` directive, a `class Math` definition, and a public method `add(int num1, int num2)`. The second screenshot shows the `Math.cpp` source file. It includes `"Math.h"` and implements the `add` method by returning the sum of `num1` and `num2`. Both files are part of a project named `MyProgram` in the `(Global Scope)`.

```
1 #pragma once
2
3 class Math
4 {
5 public:
6     int add(int num1, int num2);
7 };
8
```

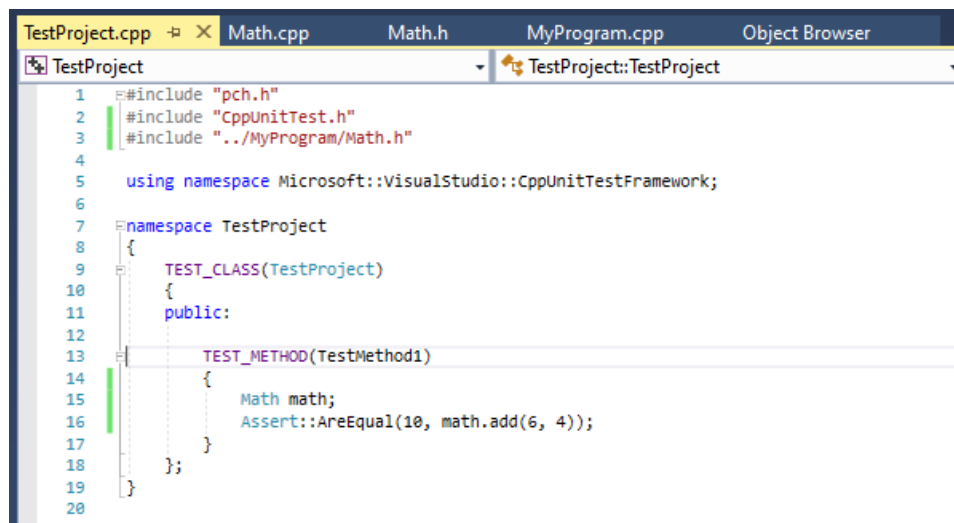
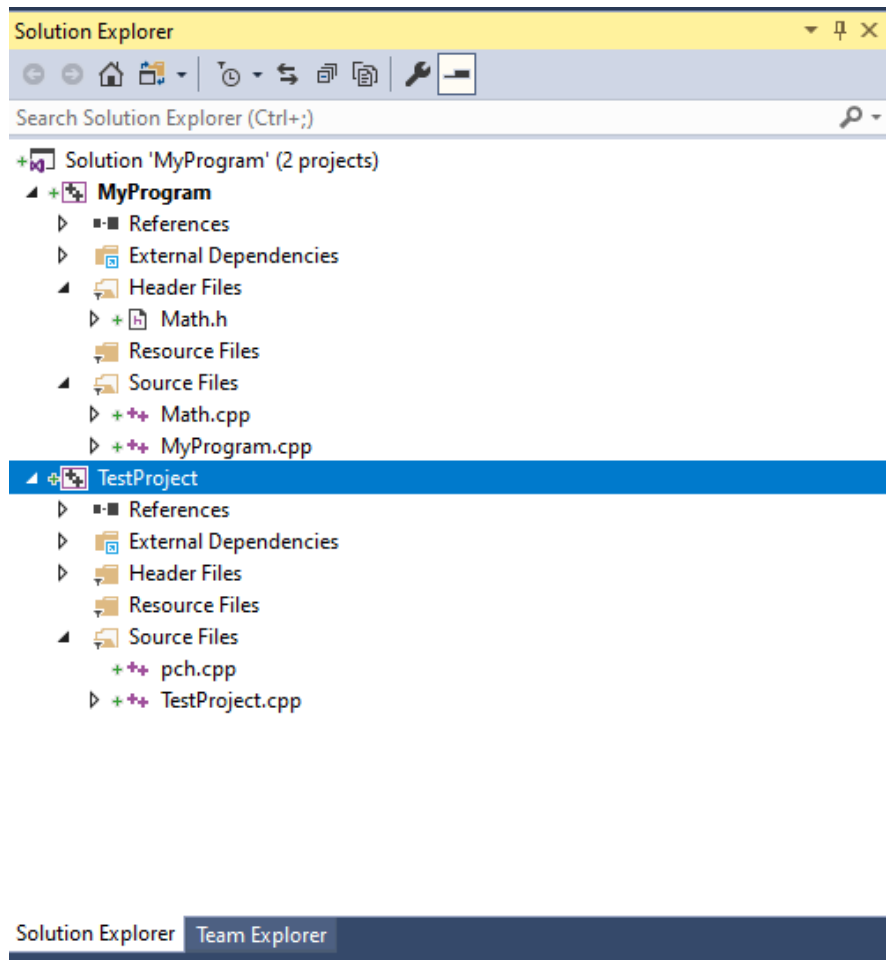
```
1 #include "Math.h"
2
3 int Math::add(int num1, int num2)
4 {
5     return num1 + num2;
6 }
7
```

To add a new test project to an existing solution, right-click on the Solution node in Solution Explorer.

In the pop-up menu, choose Add > New Project.

Select visual c++ , Test, NativeUnitTestProject

Test Project Name: TestProject



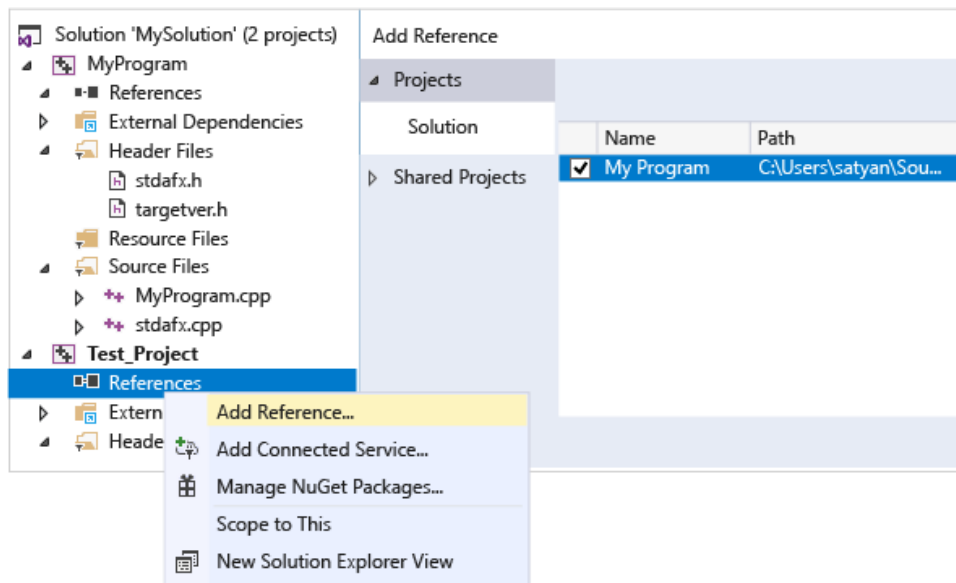
Create references to other projects in the solution

To enable access to the functions in the project under test, add a reference to the project in your test project.

Right-click on the test project node in Solution Explorer for a pop-up menu.

Choose Add > Reference.

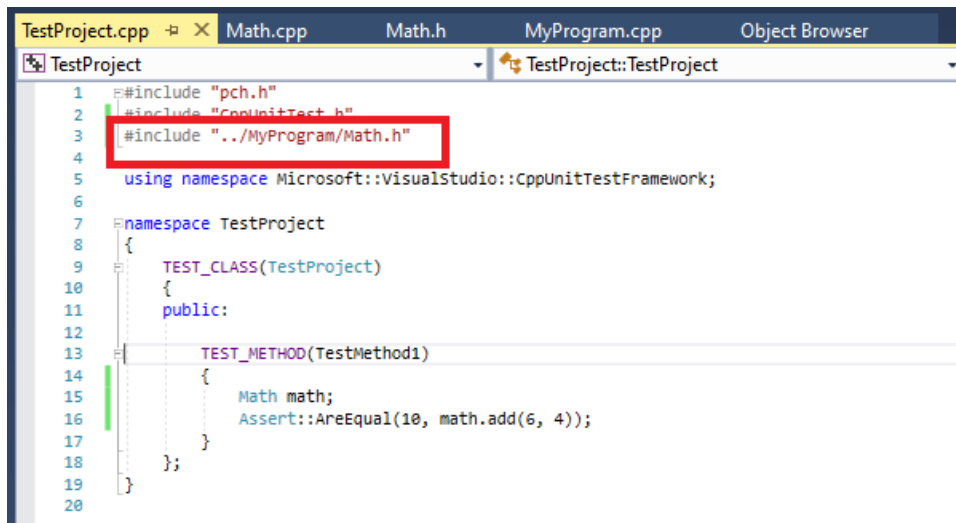
In the Add Reference dialog, choose the project(s) you want to test.



Add #include directives for header files

Next, in your unit test *.cpp* file, add an `#include` directive for any header files that declare the types and functions you want to test.

Type `#include "` and then IntelliSense will activate to help you choose. Repeat for any additional headers.



```
1 #include "pch.h"
2 #include "CppUnitTest.h"
3 #include "../MyProgram/Math.h"
4
5 using namespace Microsoft::VisualStudio::CppUnitTestFramework;
6
7 namespace TestProject
8 {
9     TEST_CLASS(TestProject)
10     {
11     public:
12
13         TEST_METHOD(TestMethod1)
14         {
15             Math math;
16             Assert::AreEqual(10, math.add(6, 4));
17         }
18     };
19 }
20
```

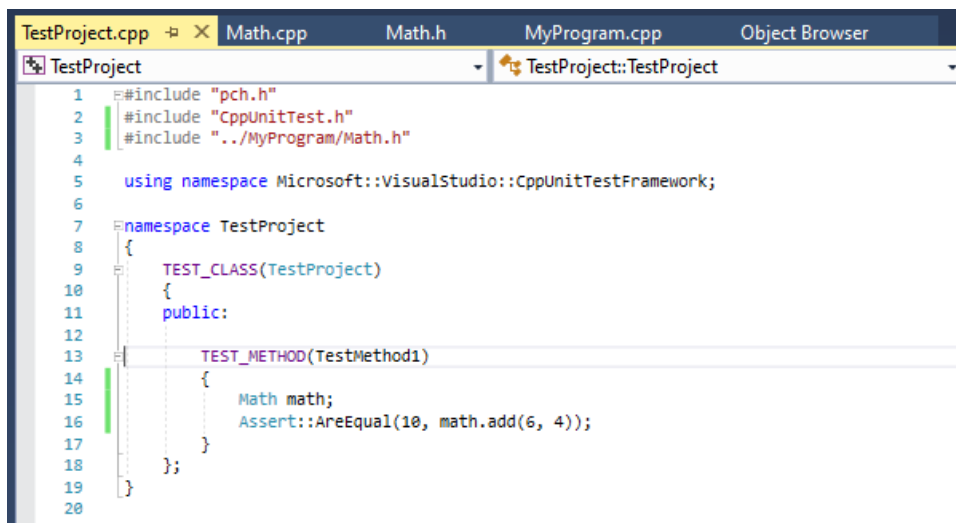
Write test methods

This section shows syntax for the Microsoft Unit Testing Framework for C/C++. It is documented here: [Microsoft.VisualStudio.TestTools.CppUnitTestFramework API reference](#).

The *.cpp* file in your test project has a stub class and method defined for you.

They show an example of how to write test code.

The signatures use the TEST_CLASS and TEST_METHOD macros, which make the methods discoverable from the Test Explorer window.



```
1 #include "pch.h"
2 #include "CppUnitTest.h"
3 #include "../MyProgram/Math.h"
4
5 using namespace Microsoft::VisualStudio::CppUnitTestFramework;
6
7 namespace TestProject
8 {
9     TEST_CLASS(TestProject)
10     {
11     public:
12
13         TEST_METHOD(TestMethod1)
14         {
15             Math math;
16             Assert::AreEqual(10, math.add(6, 4));
17         }
18     };
19 }
20
```

TEST_CLASS and TEST_METHOD are part of the [Microsoft Native Test Framework](#).

A TEST_METHOD returns void.

To produce a test result, use the static methods in the Assert class to test actual results against what is expected.

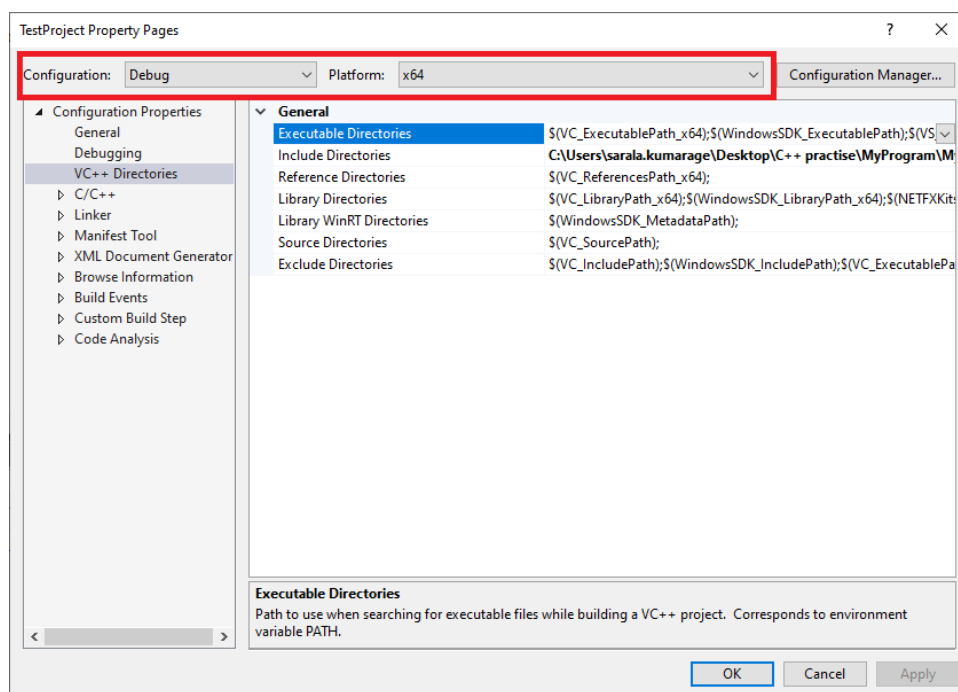
Link to object or library files

If the test code doesn't export the functions that you want to test, you can add the output .obj or .lib files to the dependencies of the test project.

Modify the test project's properties to include the headers and library or object files that are required for unit testing.

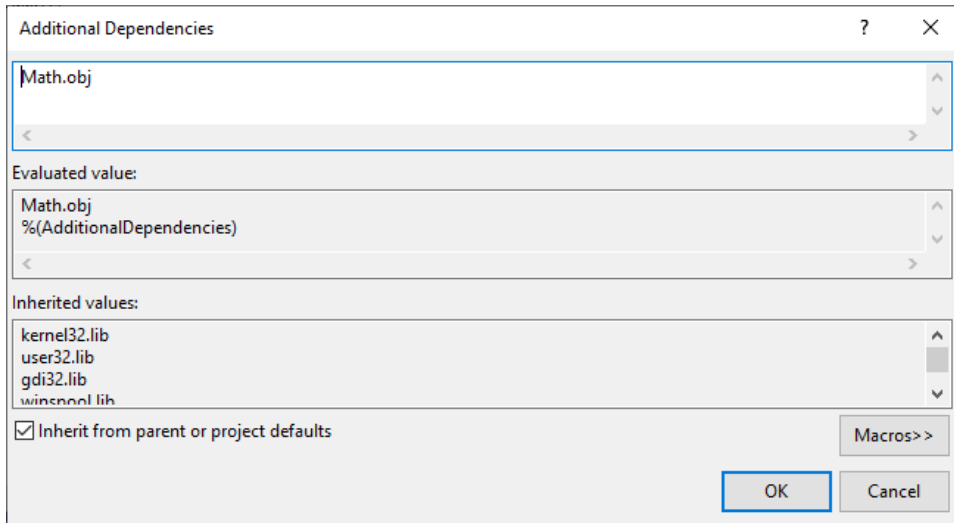
1. Solution Explorer, on the shortcut menu of the test project, choose Properties. The project properties window opens.

Set the project configuration properties accordingly. (should match the solution configurations)



2. Select the Configuration Properties > Linker > Input page, then select Additional Dependencies.

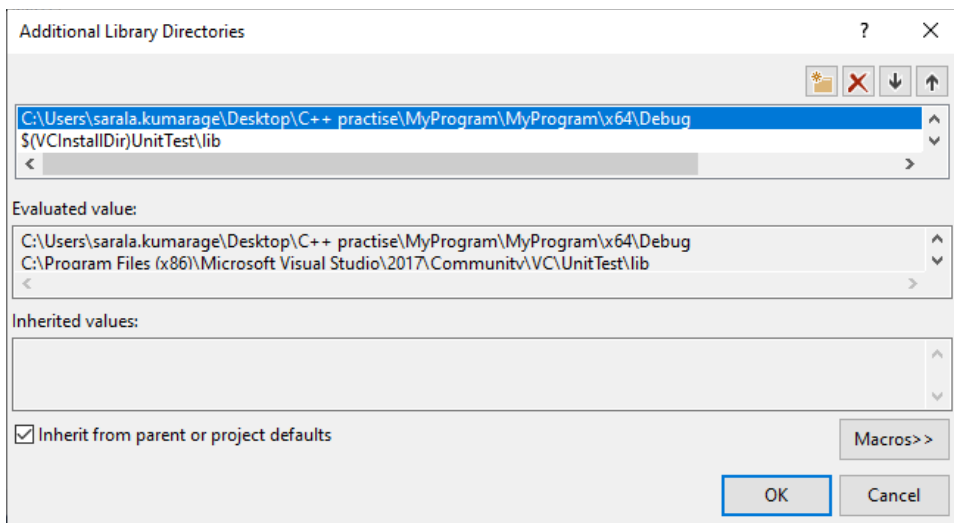
Choose Edit, and add the names of the .obj or .lib files. Don't use the full path names.



Ex: Math.obj

3. Select the Configuration Properties > Linker > General page, then select Additional Library Directories.

Choose Edit, and add the directory path of the .obj or .lib files. The path is typically within the build folder of the project under test.

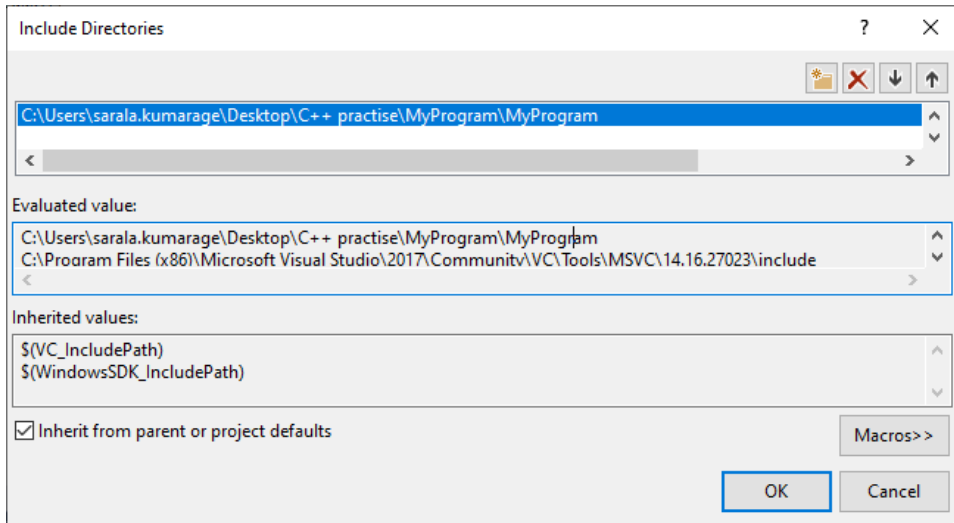


Ex:

C:\Users\sarala.kumara\Desktop\C++ practise\MyProgram\MyProgram\x64\Debug

4. Select the Configuration Properties > VC++ Directories page, then select Include Directories.

Choose Edit, and then add the header directory of the project under test.

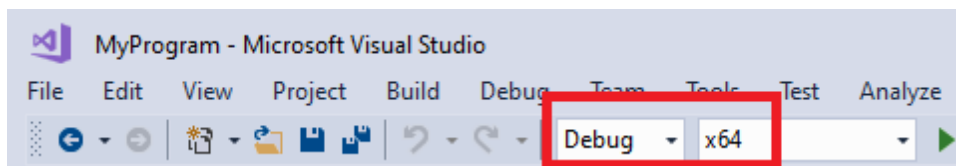


Ex:

C:\Users\sarala.kumarage\Desktop\C++ practise\MyProgram\MyProgram

Build the project

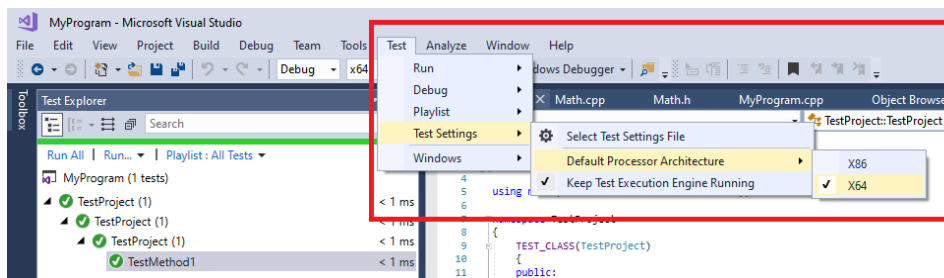
Set the solution configurations and solution platform accordingly



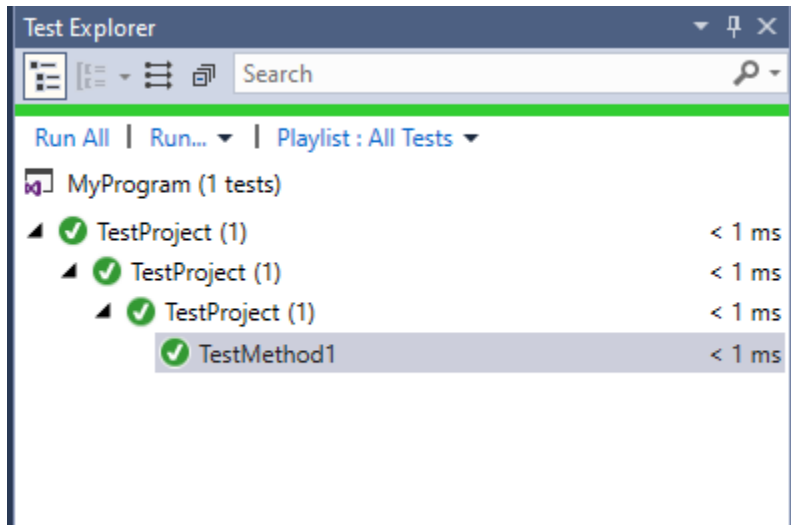
Build the solution without errors

Run the tests

Before running the test cases, choose Processor Architecture for Any CPU projects accordingly.



On the Test menu, choose Windows > Test Explorer. The following illustration shows a test project whose tests have not yet run

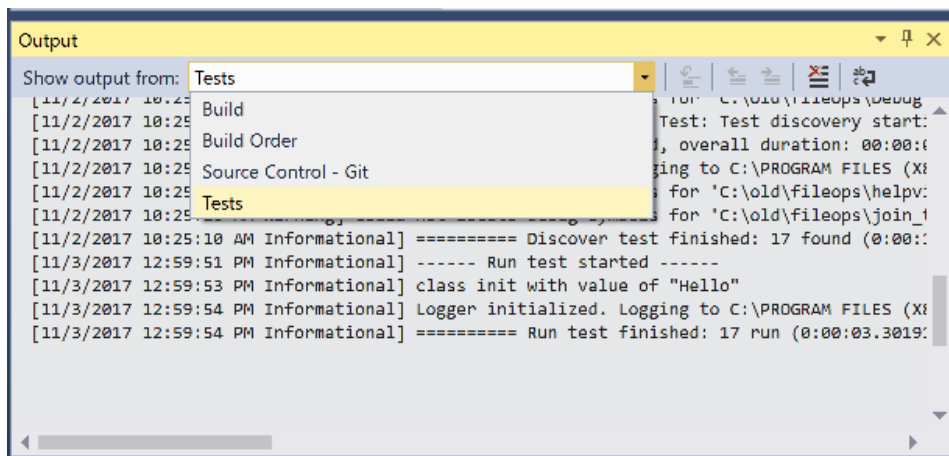


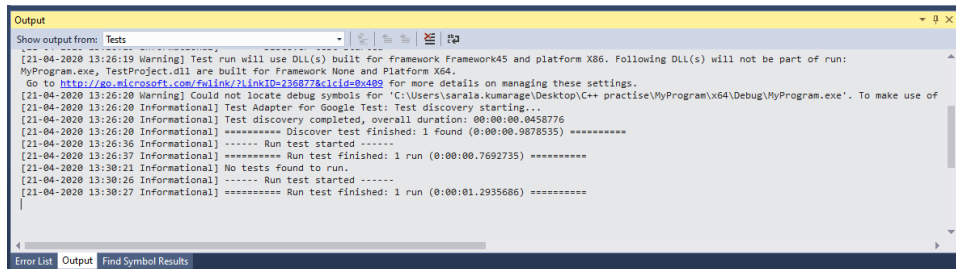
If not all your tests are visible in the window, build the test project by right-clicking its node in Solution Explorer and choosing Build or Rebuild.

In Test Explorer, choose Run All, or select the specific tests you want to run.

Test output

In the Output Window choose Tests in the drop-down to view messages written out by the Logger class.





```
Output
Show output from: Tests
[21-04-2020 13:26:19 Warning] Test run will use DLL(s) built for framework Framework45 and platform X86. Following DLL(s) will not be part of run:
MyProgram.exe, TestProject.dll are built for Framework None and Platform X64.
Go to https://go.microsoft.com/fwlink/?LinkId=234887&iclsid=9d49 for more details on managing these settings.
[21-04-2020 13:26:20 Informational] Test Adapter for Google Test: Test discovery starting...
[21-04-2020 13:26:20 Informational] Test discovery completed, overall duration: 00:00:00.0458776
[21-04-2020 13:26:20 Informational] ===== Discover test finished: 1 found (0:00:00.9878535) =====
[21-04-2020 13:26:36 Informational] ----- Run test started -----
[21-04-2020 13:26:37 Informational] ===== Run test finished: 1 run (0:00:00.7692735) =====
[21-04-2020 13:30:21 Informational] No tests found to run.
[21-04-2020 13:30:26 Informational] ----- Run test started -----
[21-04-2020 13:30:27 Informational] ===== Run test finished: 1 run (0:00:01.2935686) =====
```

References

<https://docs.microsoft.com/en-us/visualstudio/test/writing-unit-tests-for-c-cpp?view=vs-2019#create-a-test-project-in-visual-studio-2019>

https://docs.microsoft.com/en-us/visualstudio/test/how-to-use-microsoft-test-framework-for-cpp?view=vs-2019#object_files

<https://docs.microsoft.com/en-us/visualstudio/test/run-a-unit-test-as-a-64-bit-process?view=vs-2019>

https://docs.microsoft.com/en-us/visualstudio/test/how-to-use-microsoft-test-framework-for-cpp?view=vs-2019#object_files

API reference

<https://docs.microsoft.com/en-us/visualstudio/test/microsoft-visualstudio-testtools-cppunittestframework-api-reference?view=vs-2019>