

## Object Oriented Programming Lab

### Week 1



### Lab Objectives

Visual Studio Installation and execution

Google Test on Visual Studio

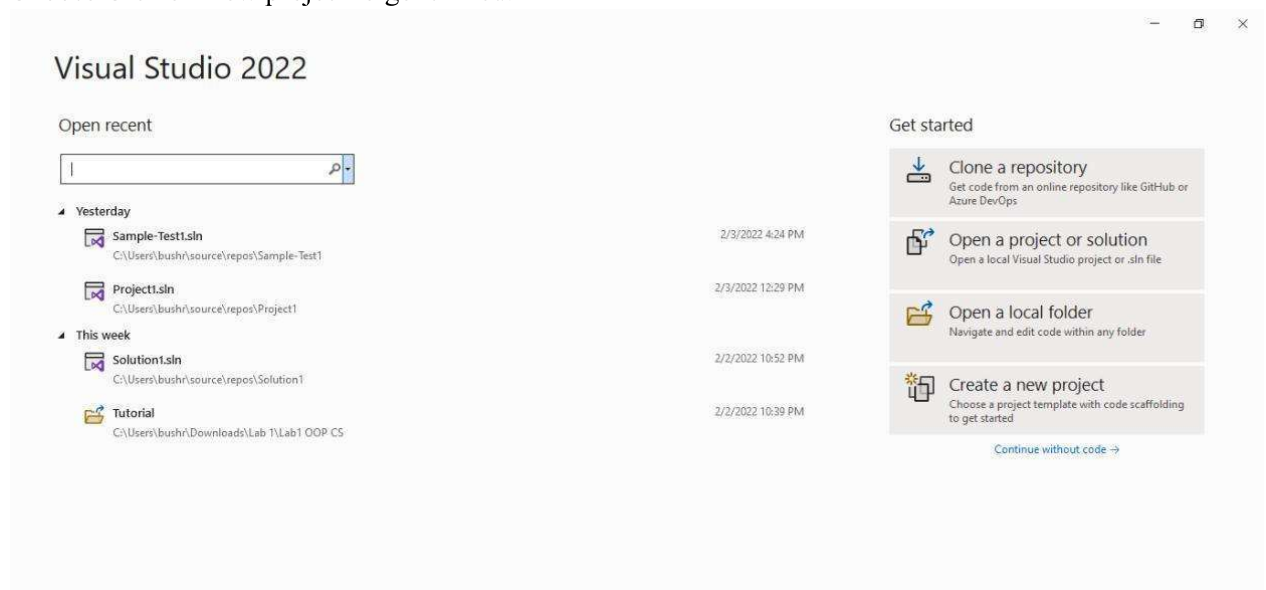
Debugging in Visual Studio

# Getting started with Visual Studio

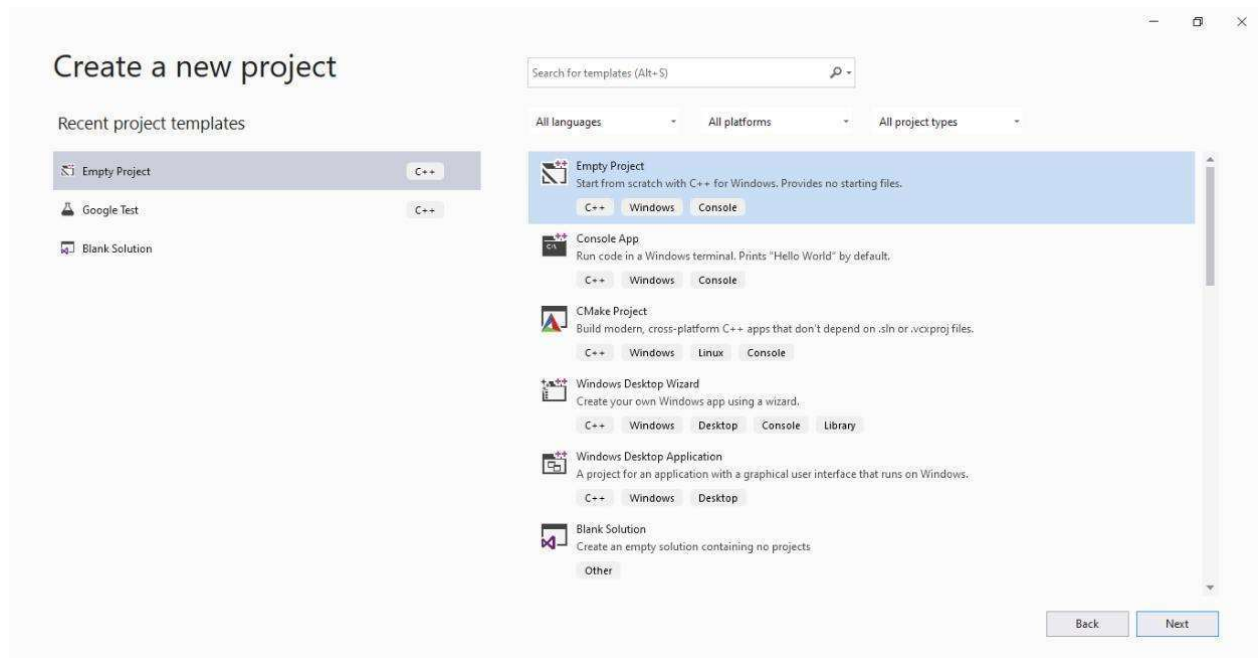
The usual starting point for a C++ programmer is a "Hello, world!" application that runs on the command line. That's what you'll create first in Visual Studio, and then we'll move on to something more challenging



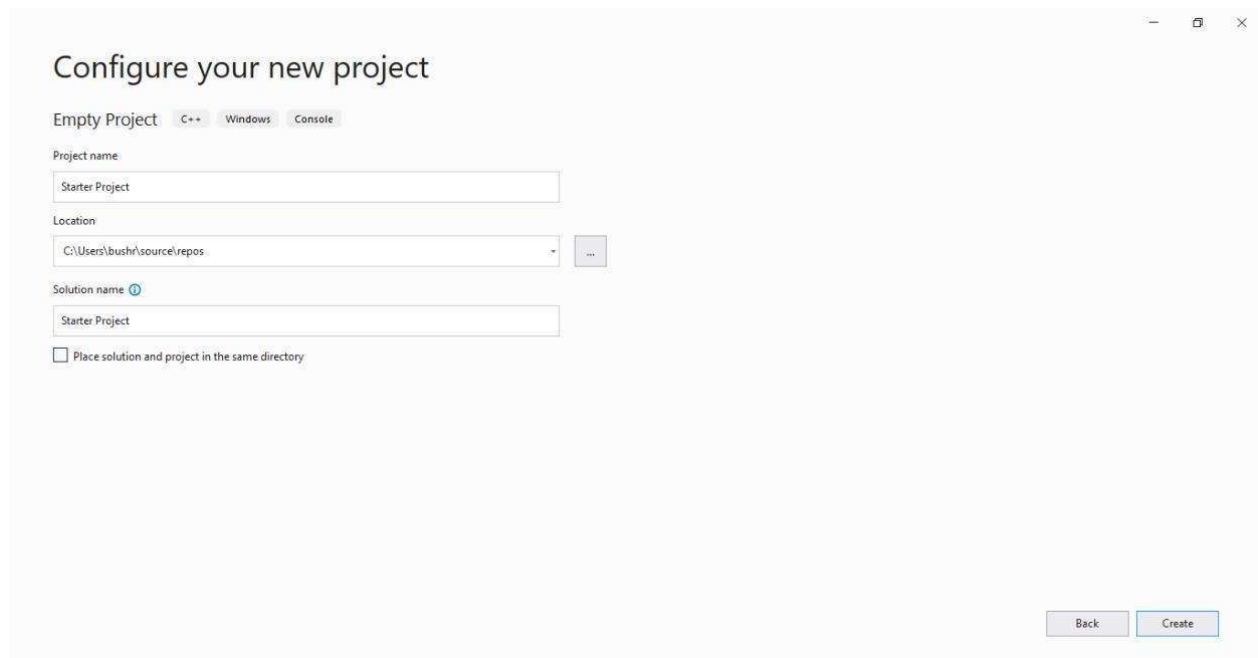
1. Choose Create a new project to get started.



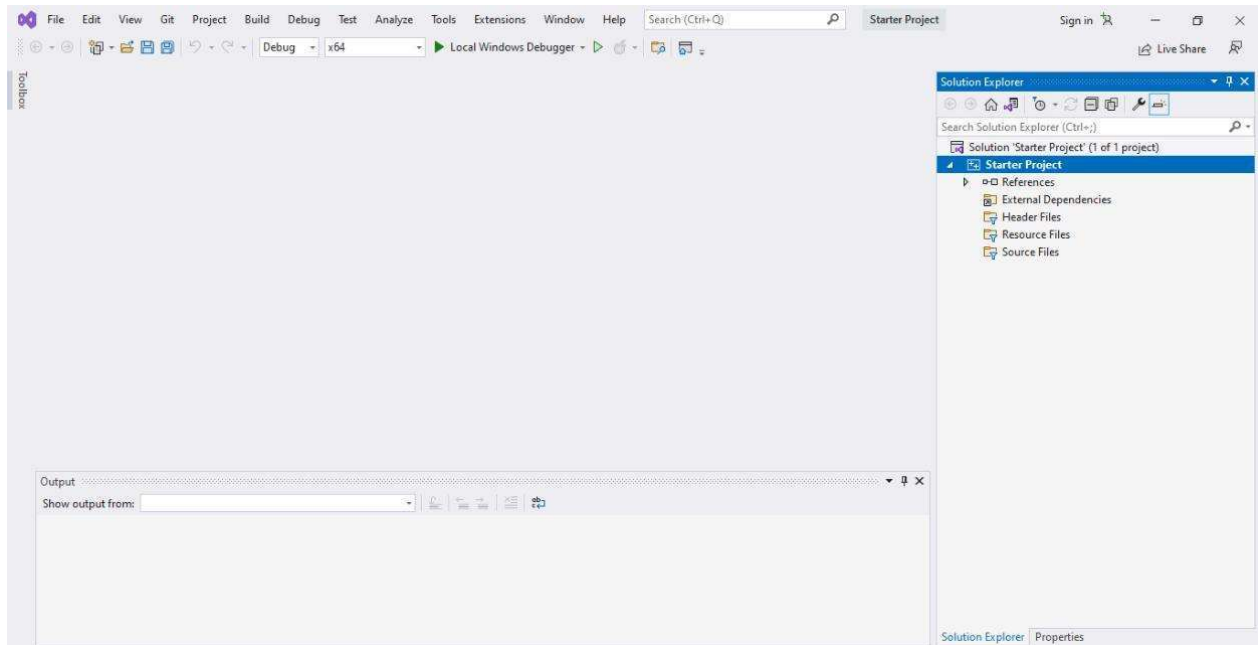
2. In the list of project templates, choose Empty Project, then choose Next.



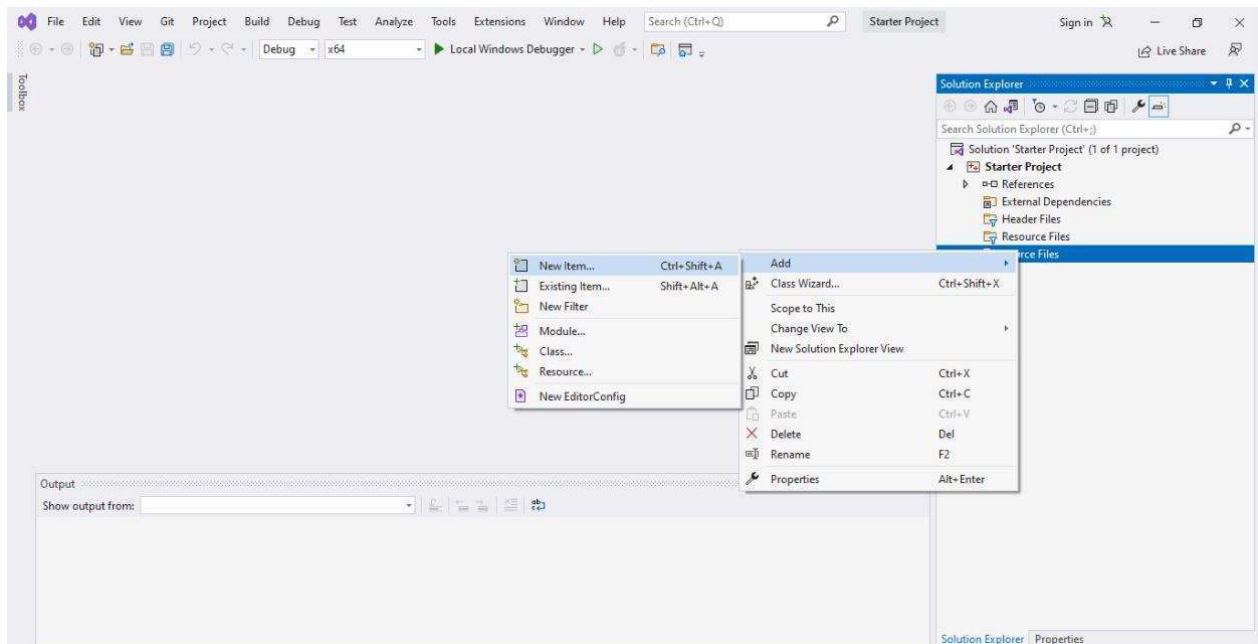
3. In the Configure your new project dialog box, select the Project Name edit box, name your new project Starter Project, then choose to Create.



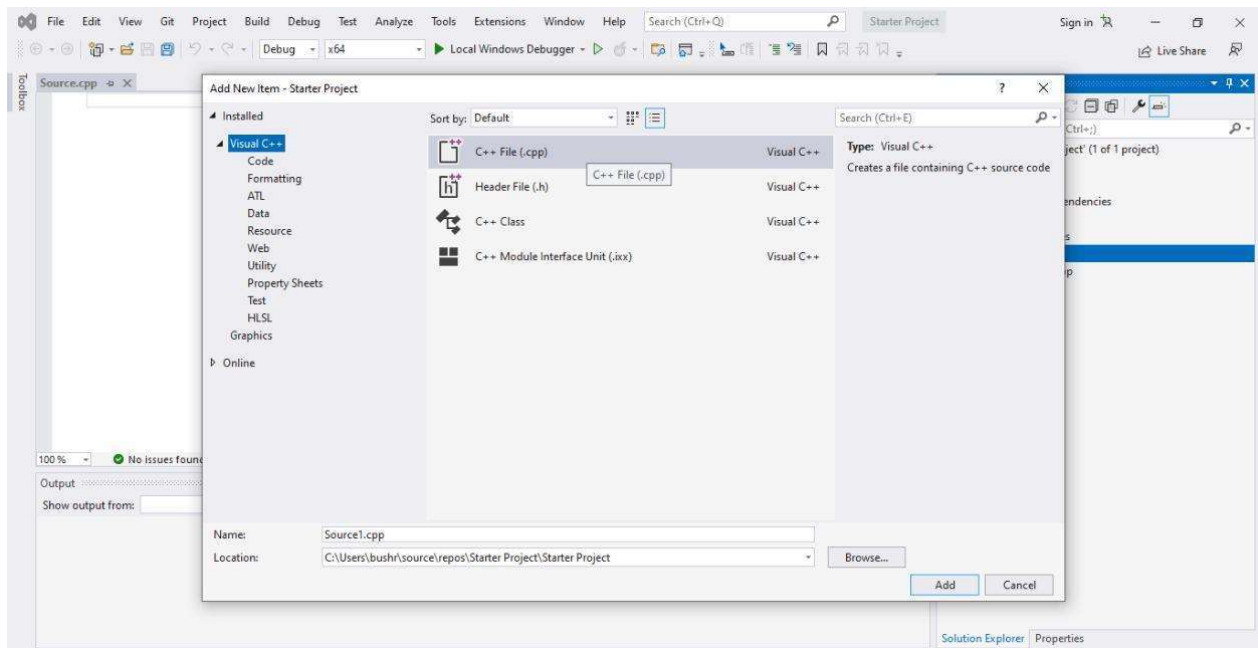
4. An empty C++ Windows console application gets created. Console applications use a Windows console window to display output and accept user input. In Visual Studio, an empty project will be created.



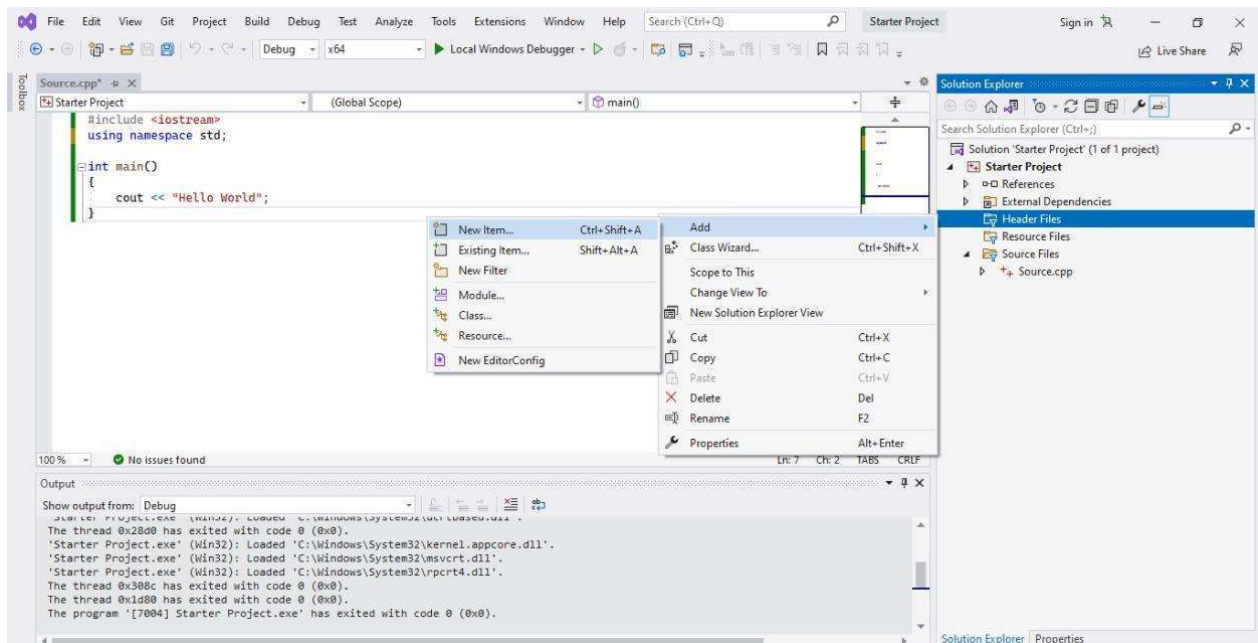
5. Right-click on Source Files and add a new item.



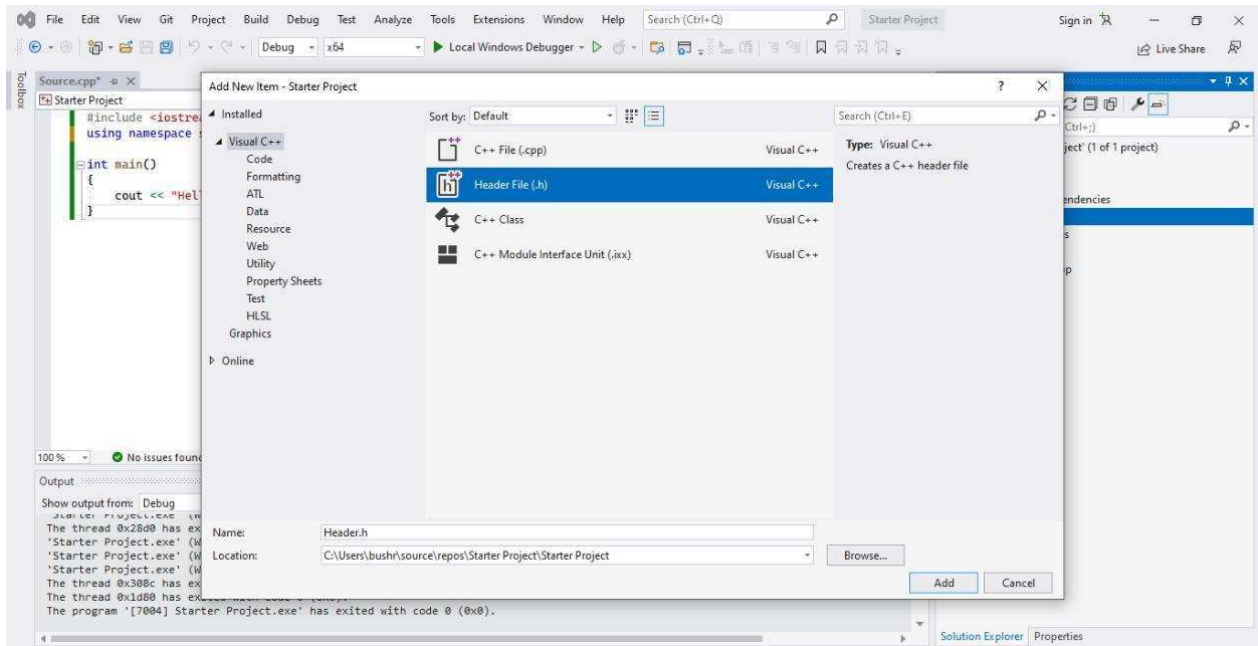
## 6. Create .cpp file and click Add



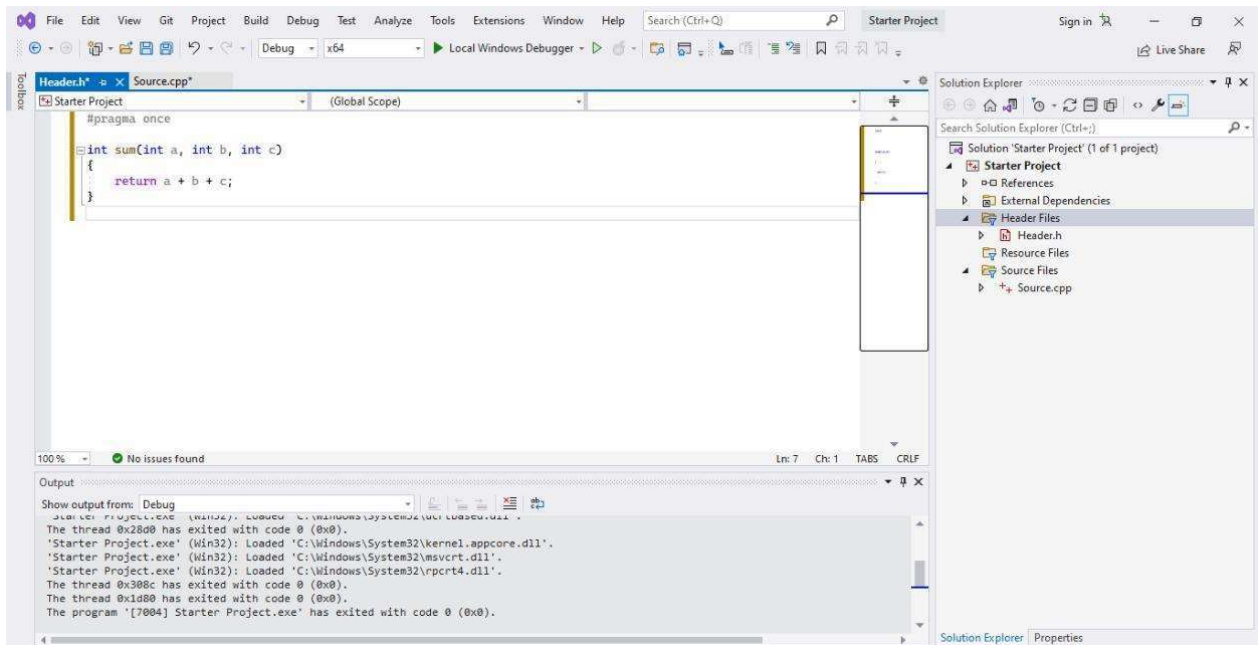
## 7. Right-click on Header Files and add a new item.



## 8. Select header file and click add

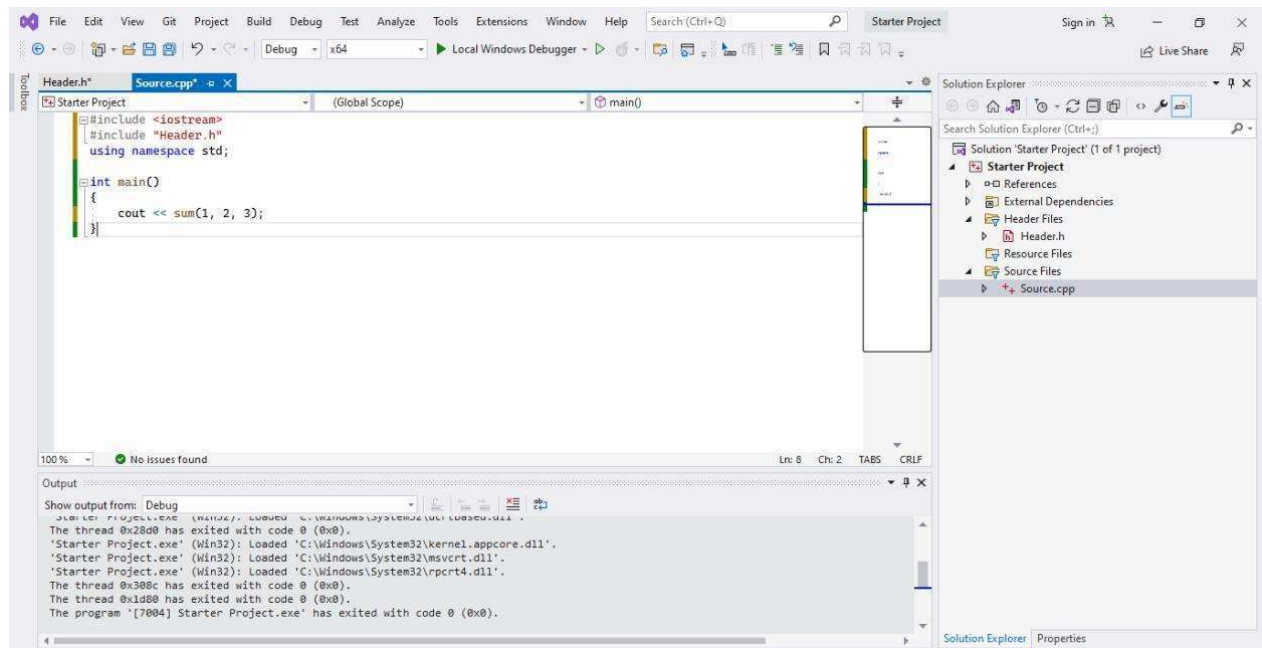


## 9. This is where you are going to write all your functions. Write sum() function in header file

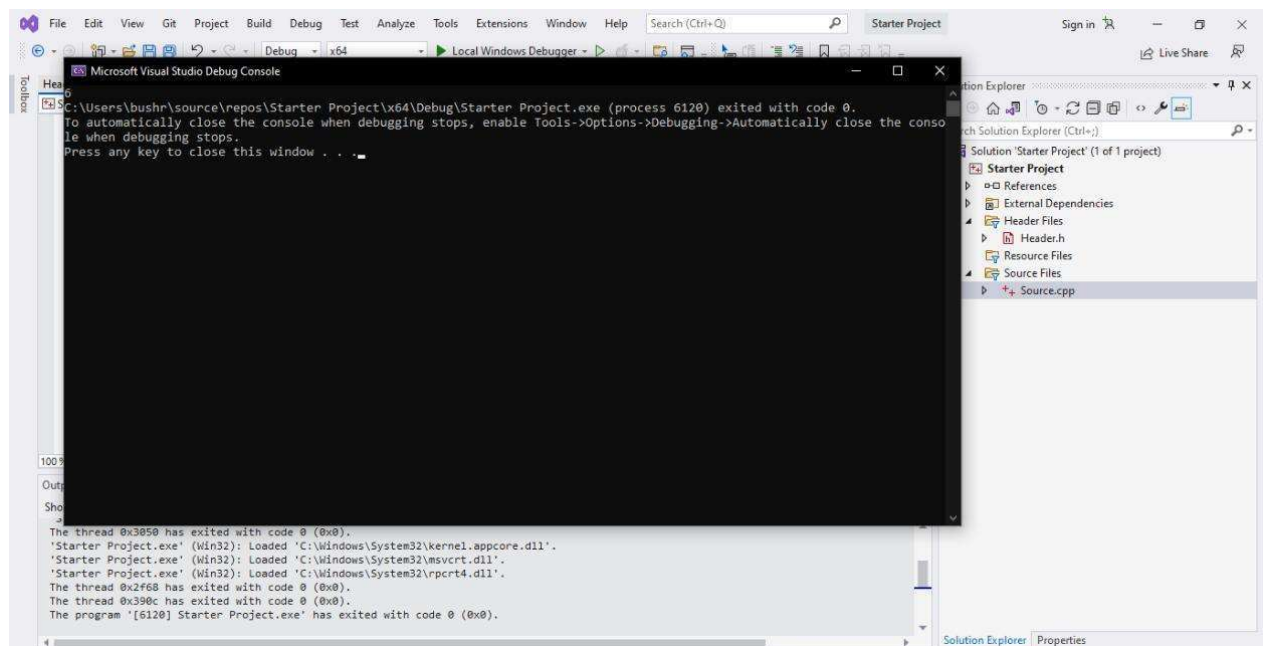




10. You can use the functions written in the header file by including the file in Source.cpp.



11. Compile and run the code using the Local Windows Debugger option from the quick access toolbar followed by the green play button.



# Getting started with Gtest

Google test is a framework for writing C++ unit tests.

- ☐ When using googletest, you start by writing *assertions*, which are statements that check whether a condition is true.
- ☐ **Tests** use assertions to verify the tested code's behavior. If a test crashes or has a failed assertion, then it *fails*; otherwise it *succeeds*.
- ☐ A **test suite** contains one or many tests. You should group your tests into test suites that reflect the structure of the tested code.
- ☐ A **test program** can contain multiple test suites.

This is how a test suite looks like,

```
TEST_F(TestFixtureName, TestName) {  
    ... test body ...  
}
```

Example

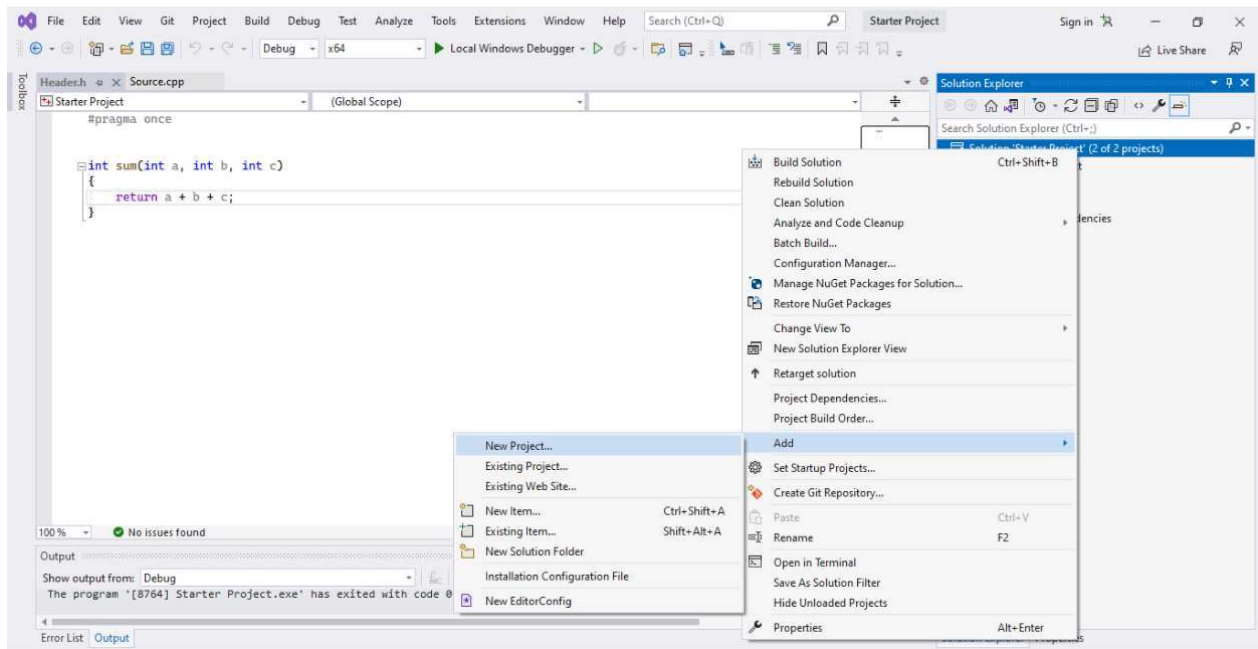
```
// Tests factorial of positive numbers.  
TEST(FactorialTest, HandlesPositiveInput) {  
    EXPECT_EQ(Factorial(1), 1);  
    EXPECT_EQ(Factorial(2), 2);  
    EXPECT_EQ(Factorial(3), 6);  
    EXPECT_EQ(Factorial(8), 40320);  
}
```

## Visual Studio

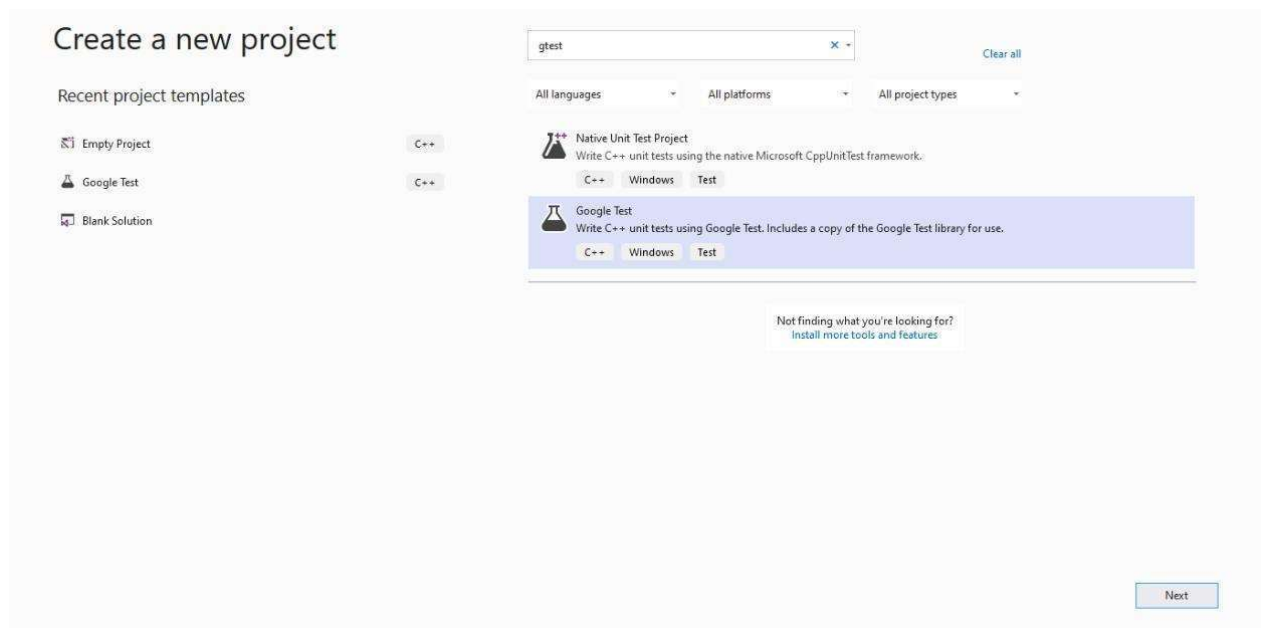
In Visual Studio 2017 and later, Google Test is integrated into the Visual Studio IDE as a default component of the Desktop Development with C++ workload.

1. In Solution Explorer, right-click on the solution node and choose Add > New Project. Add a Google Test project in Visual Studio. Click on File and create a new project

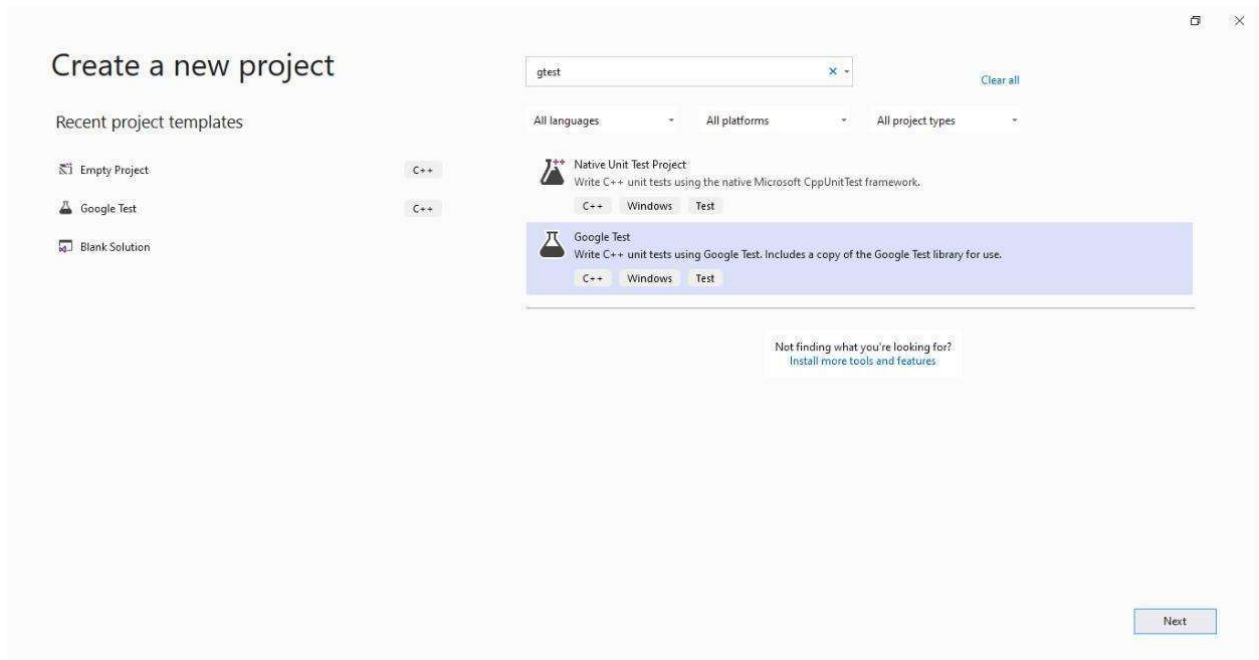




2. Search for google test and click next



3. Rename the project and click create



Create a new project

Recent project templates

- Empty Project C++
- Google Test C++
- Blank Solution

gtest Clear all

All languages: All platforms: All project types:

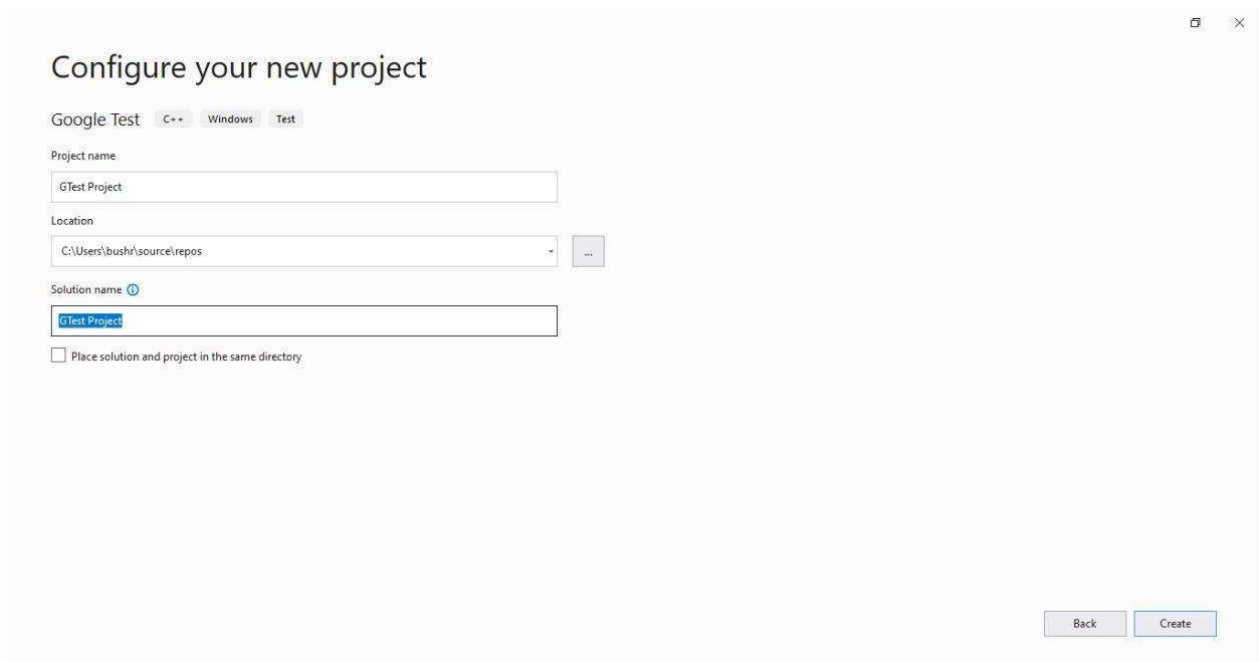
Native Unit Test Project: Write C++ unit tests using the native Microsoft CppUnitTest framework. C++ Windows Test

Google Test: Write C++ unit tests using Google Test. Includes a copy of the Google Test library for use. C++ Windows Test

Not finding what you're looking for? [Install more tools and features](#)

Next

4. Configure your project and click create



Configure your new project

Google Test C++ Windows Test

Project name: GTest Project

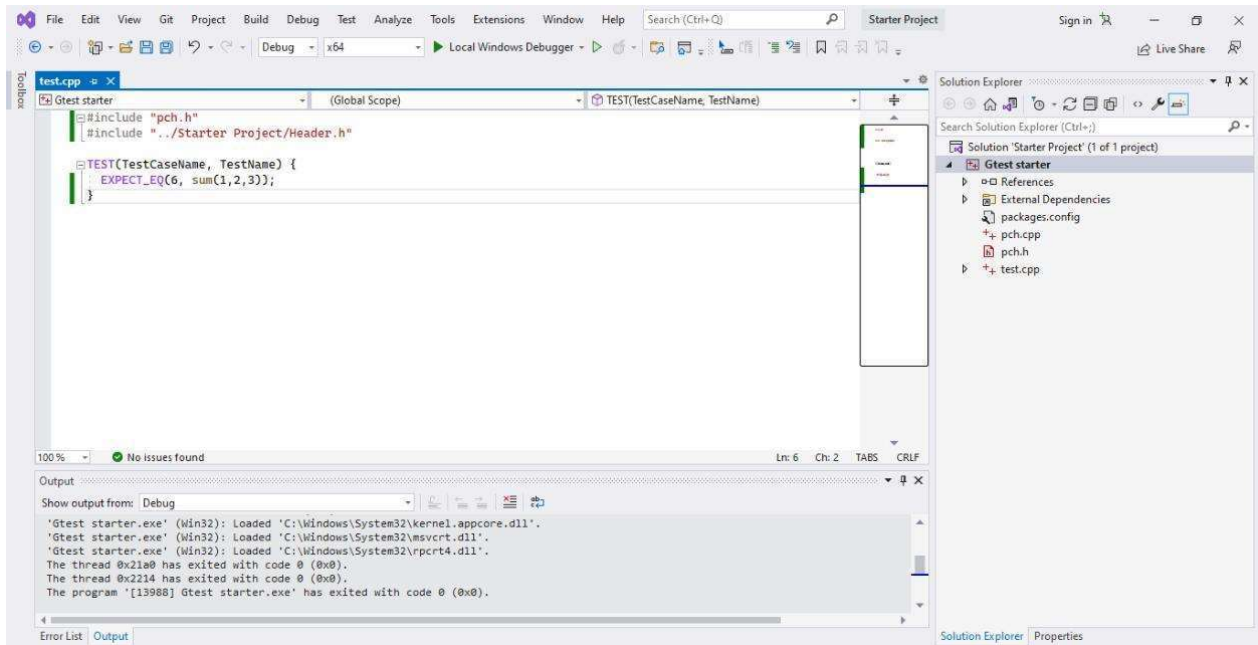
Location: C:\Users\bushri\source\repos ...

Solution name ⓘ: GTest Project

☐ Place solution and project in the same directory

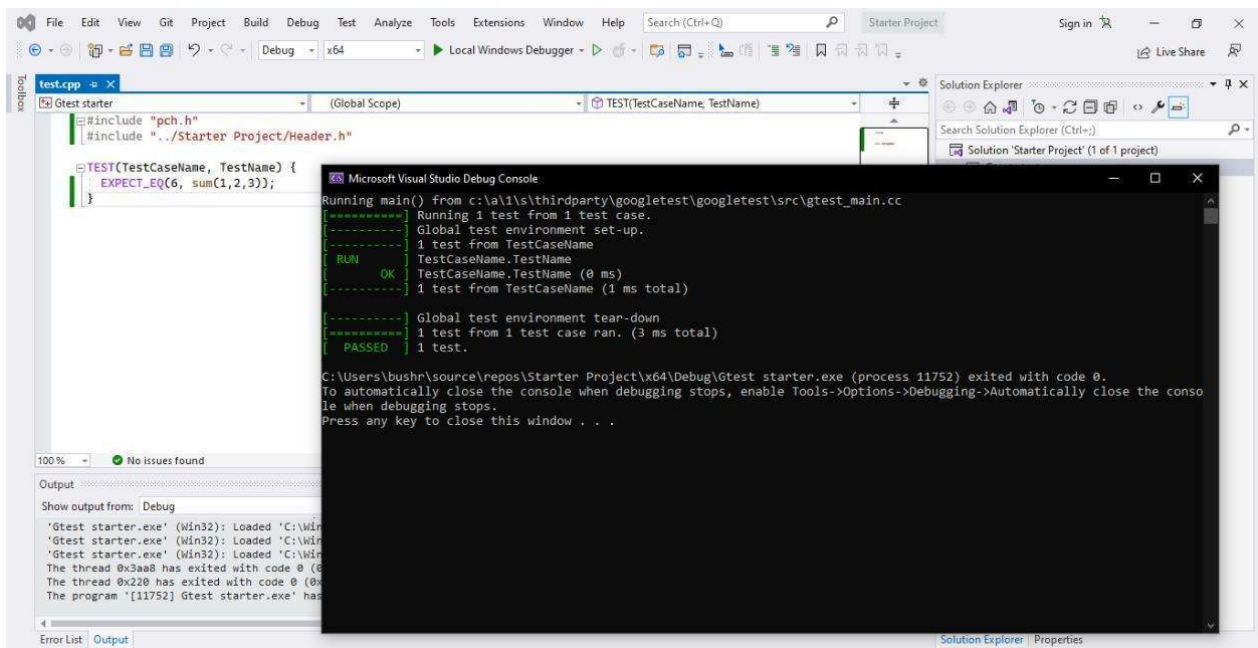
Back Create

5. You are now ready to write and run Google Tests. Include the header file you want to test and write the test case

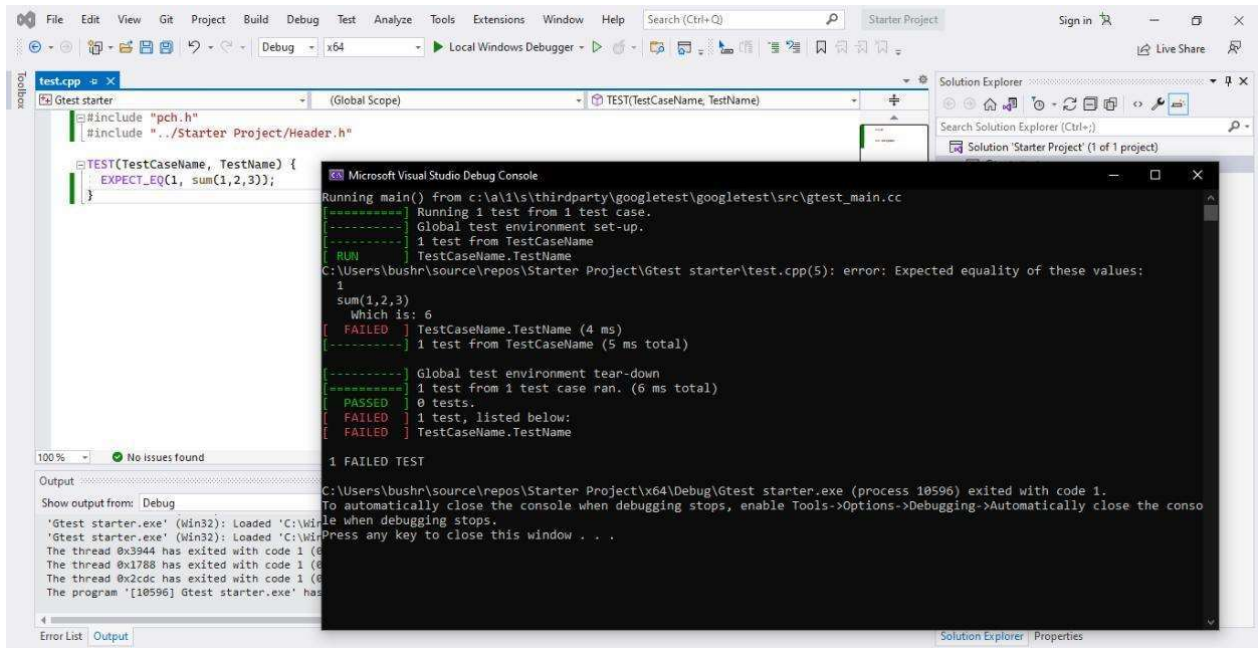


6. Compile and run the code using the Local Windows Debugger option from the quick access toolbar followed by the green play button

a. Passed Test Case



## b. Failed test case



The screenshot shows the Visual Studio IDE with a C++ project named 'Gtest starter'. The code in `test.cpp` defines a test case `TEST(TestCaseName, TestName)` that expects the sum of 1, 2, and 3 to be 1. The debug console shows the test execution results, indicating a failure.

```
#include "pch.h"
#include "../Starter Project/Header.h"

TEST(TestCaseName, TestName) {
    EXPECT_EQ(1, sum(1,2,3));
}
```

Microsoft Visual Studio Debug Console

```
Running main() from c:\a\1\thirdparty\googletest\googletest\src\gtest_main.cc
----- Running 1 test from 1 test case.
----- Global test environment set-up.
----- 1 test from TestCaseName
RUN
    TestCaseName.TestName
C:\Users\bushn\source\repos\Starter Project\Gtest starter\test.cpp(5): error: Expected equality of these values:
1
sum(1,2,3)
  Which is: 6
[ FAILED ] TestCaseName.TestName (4 ms)
----- 1 test from TestCaseName (5 ms total)
----- Global test environment tear-down
----- 1 test from 1 test case ran. (6 ms total)
----- PASSED: 0 tests.
----- FAILED: 1 test, listed below:
----- [ FAILED ] TestCaseName.TestName

1 FAILED TEST

C:\Users\bushn\source\repos\Starter Project\x64\Debug\Gtest starter.exe (process 10596) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

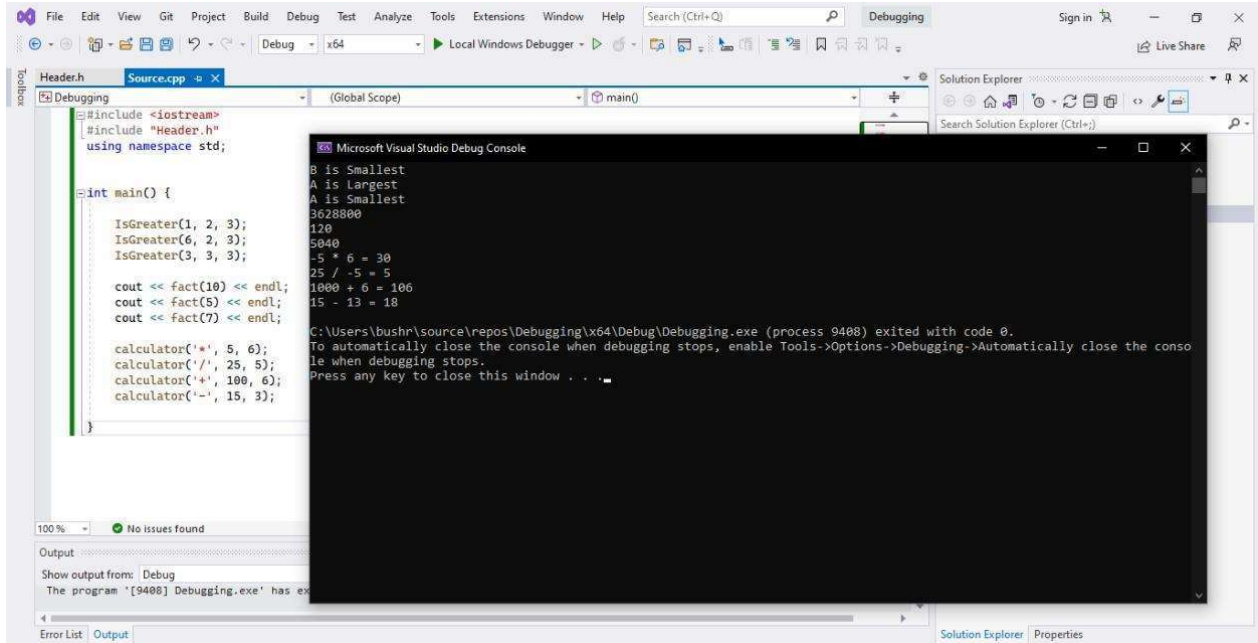
Output

```
'Gtest starter.exe' (Win32): Loaded 'C:\Windows\System32\ntdll.dll'.
'Gtest starter.exe' (Win32): Loaded 'C:\Windows\System32\kernel32.dll'.
The thread 0x3944 has exited with code 1 (0).
The thread 0x1788 has exited with code 1 (0).
The thread 0x2cdc has exited with code 1 (0).
The program '[10596] Gtest starter.exe' has exited with code 1 (0).
```

# Debugging in Visual Studio

When you debug your app, it usually means that you are running your application with the debugger attached. When you do this, the debugger provides many ways to see what your code is doing while it runs. You can step through your code and look at the values stored in variables, you can set watches on variables to see when values change, you can examine the execution path of your code, see whether a branch of code is running, and so on.

1. Download **debugging\_01.cpp** from classroom
2. Now open Visual Studio and create a new C++ project
3. Name your project as **DebugExercise**
4. Create a **source file** and copy the content of the provided files to your created Project files.
5. Run the code, you will see the following output



The screenshot shows the Visual Studio IDE with the following components:

- Source Code (Source.cpp):**

```
#include <iostream>
#include "Header.h"
using namespace std;

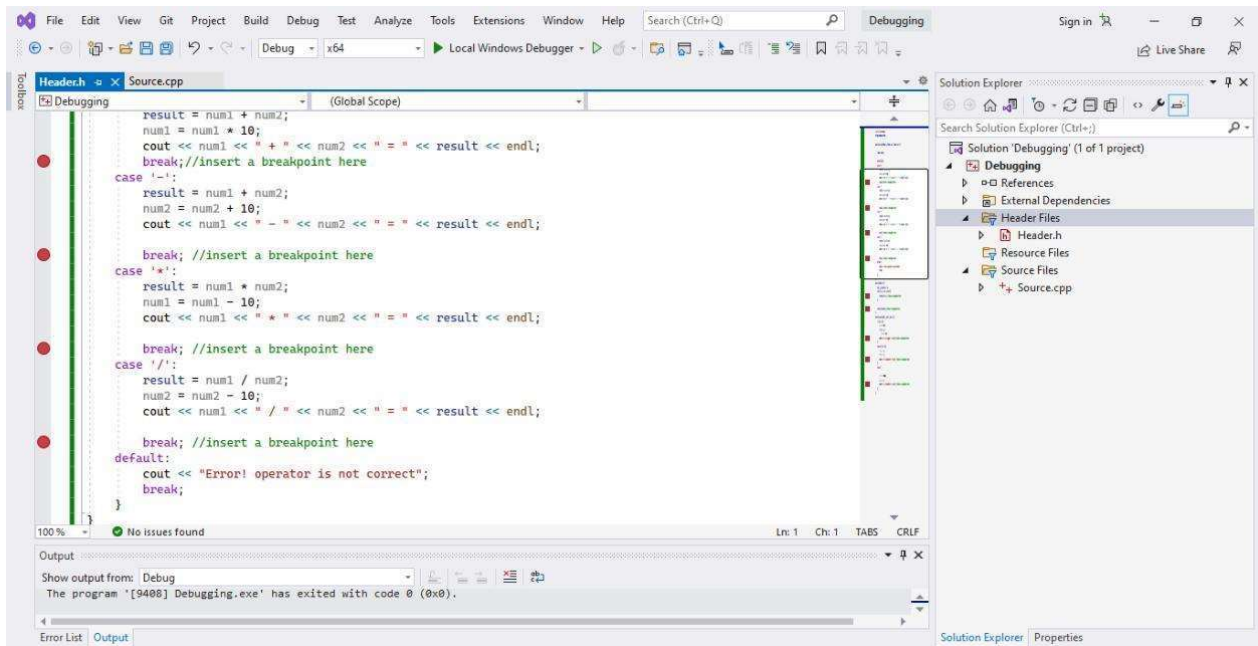
int main() {
    IsGreater(1, 2, 3);
    IsGreater(6, 2, 3);
    IsGreater(3, 3, 3);

    cout << fact(10) << endl;
    cout << fact(5) << endl;
    cout << fact(7) << endl;

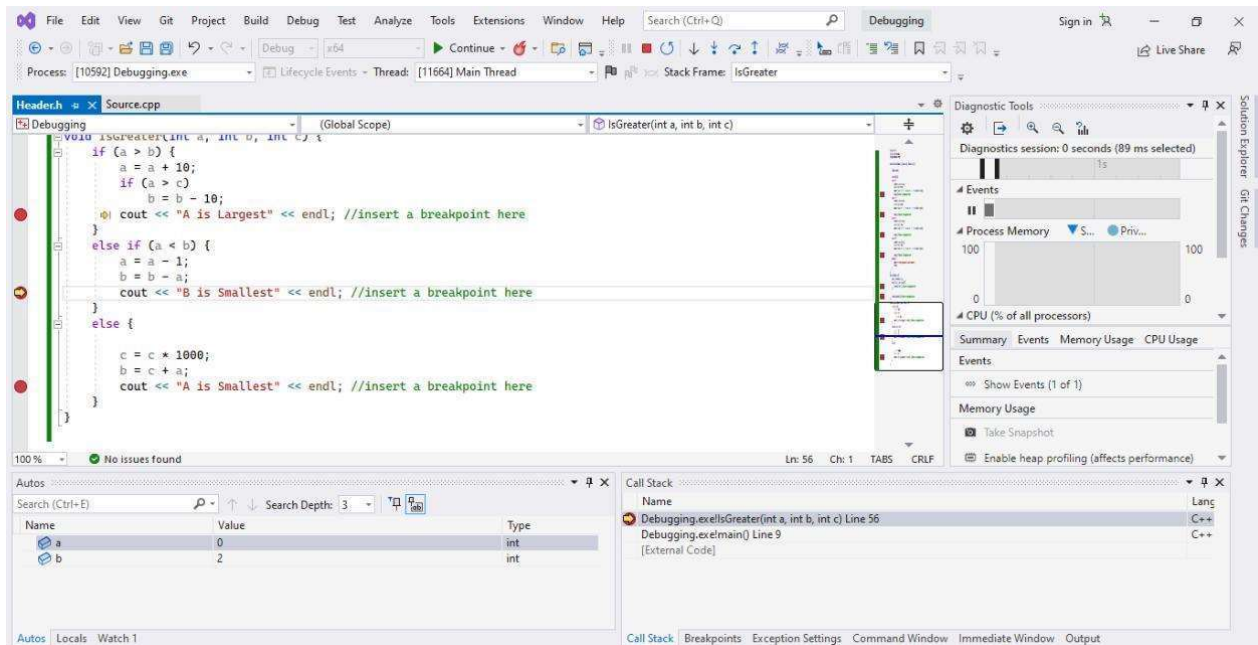
    calculator('*', 5, 6);
    calculator('/', 25, 5);
    calculator('+', 100, 6);
    calculator('-', 15, 3);
}
```
- Debug Console:** Displays the output of the program:

```
8 is Smallest
A is Largest
A is Smallest
3628800
120
5040
-5 * 6 = 30
25 / -5 = 5
1000 + 6 = 106
15 - 13 = 18
C:\Users\bushn\source\repos\Debugging\x64\Debug\Debugging.exe (process 9408) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```
- Output Window:** Shows the message: "The program '[9408] Debugging.exe' has exited with code 0 (0)."

6. Now add breakpoints in the code at mentioned lines.



7. After adding breakpoints, start debugging the project by clicking on **Debug Icon in the Quick Access toolbar**.



8. Use step in and step out to check the value of each declared variable inside all functions.



## Lab Task

1. Create a header file and write functions for Addition, Subtraction, Multiply and Division of 2 numbers and run the provided test case file “**test\_cases\_01.cpp**”.
2. Open “**debugging\_01.cpp**” file and for each function, at all given breakpoints, you need to report the values of each local variable declared.

### Submission Instructions

1. Save all .cpp files with your roll no and task number e.g. i21XXXX\_Problem01.cpp
2. Now create a new folder with name ROLLNO\_LAB\_01 e.g. i21XXXX\_LAB\_01
3. Move all of your .cpp files to this newly created directory and compress it into .zip file
4. Now you have to submit this zipped file on Google Classroom.