Visual Studio 2017 – QuickStart Guide

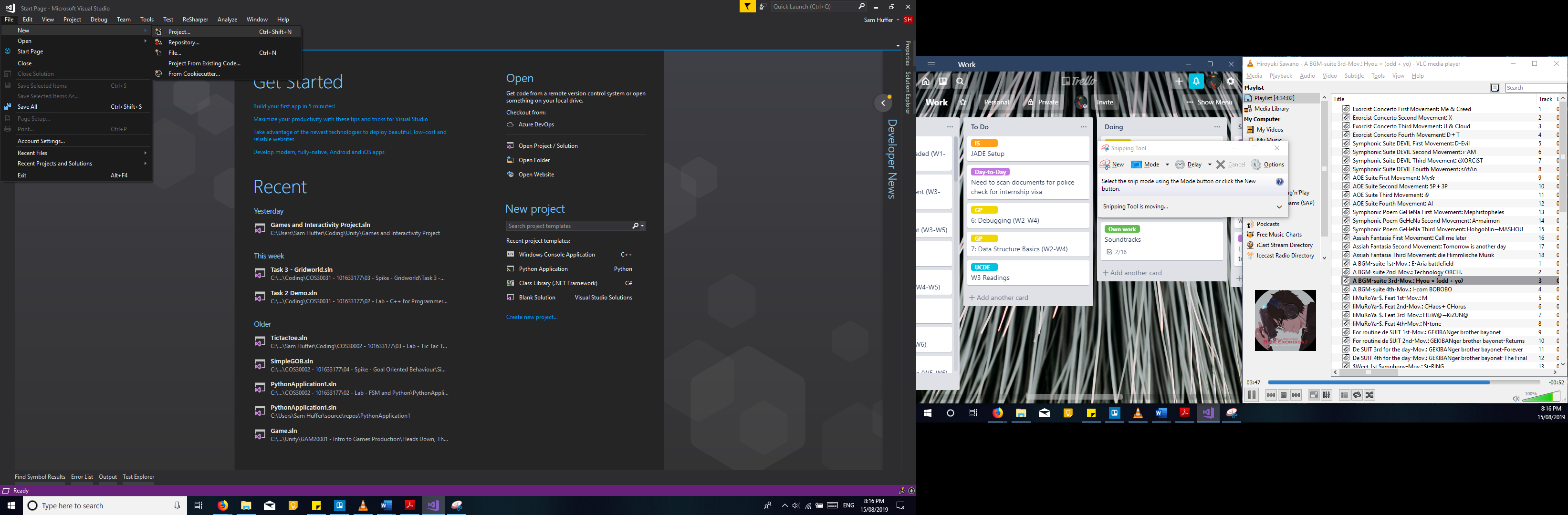
# Overview

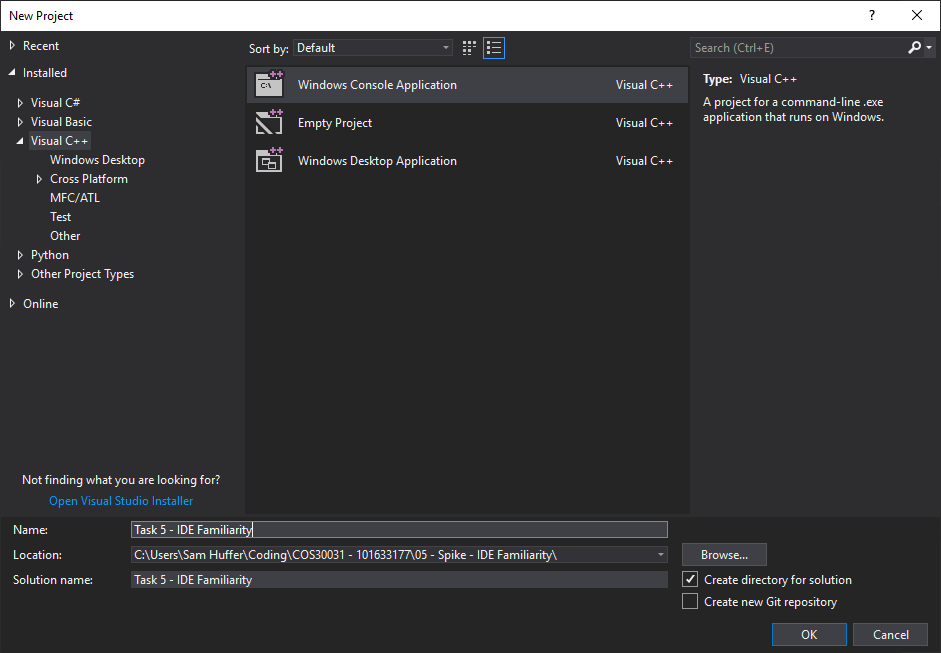
1. Project Setup
   1. Creating a new command line project
   2. Adding new files
   3. Compiling the program
   4. Running the program
2. Debugging
   1. Inserting breakpoints
   2. Run program so it stops at the breakpoint
   3. Inspecting values of variables during debugging

# 1. Project Setup

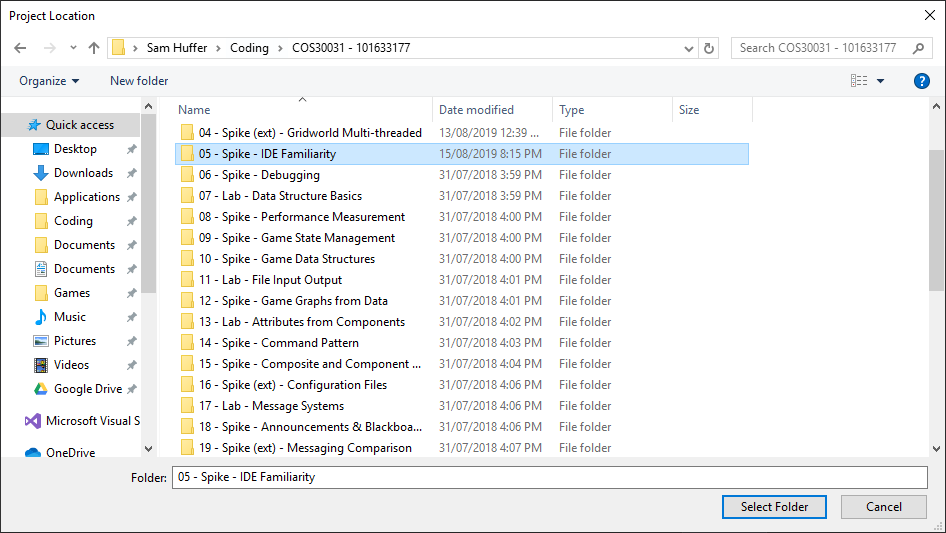
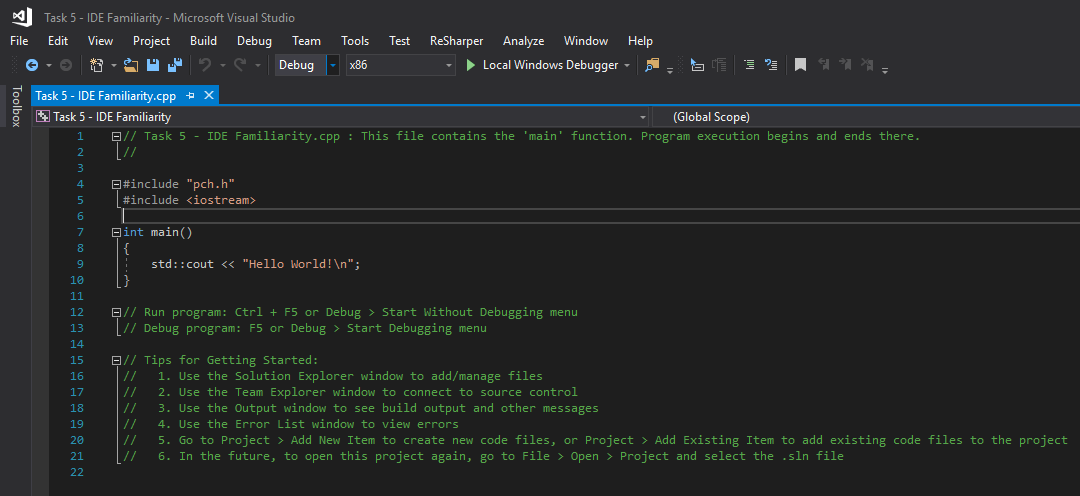
## 1.1. Creating a new command line project

Step 1: In the top-left corner, click “File > New > Project”. This will open a dialog box for creating a new project. Shortcut keys: Ctrl + Shift + N.





Step 2: When the dialog box opens, there will be a panel on the left with various project types grouped by language. Click on “Visual C++”. This will show the appropriate project types in the centre of the dialog box. The one we want is the “Windows Console Application”; click on it to select it. Down the bottom you can enter your chosen name for your solution, and choose where to store it by clicking “Browse”; this will open a second dialog box.



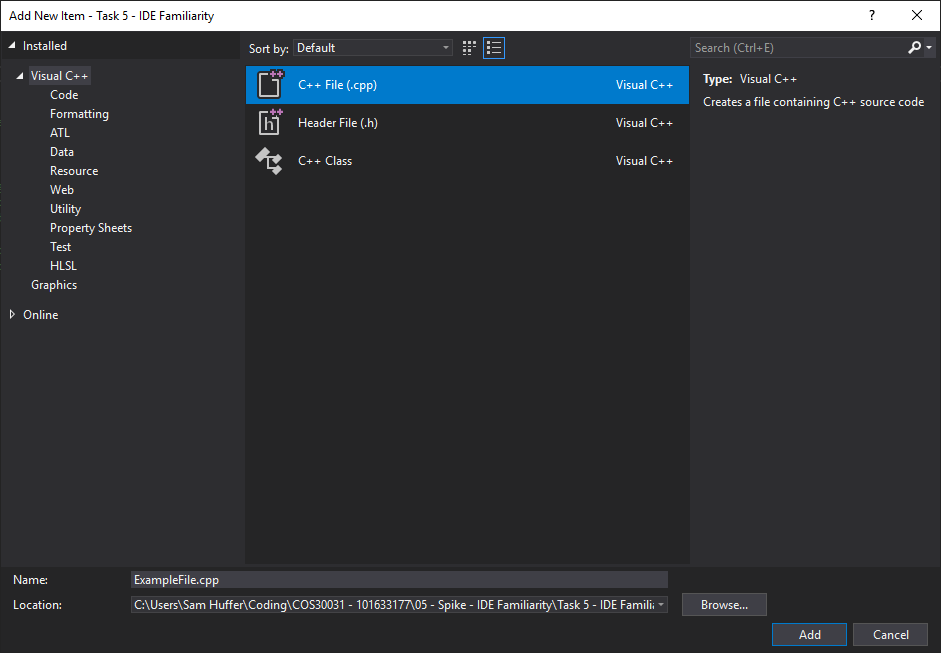
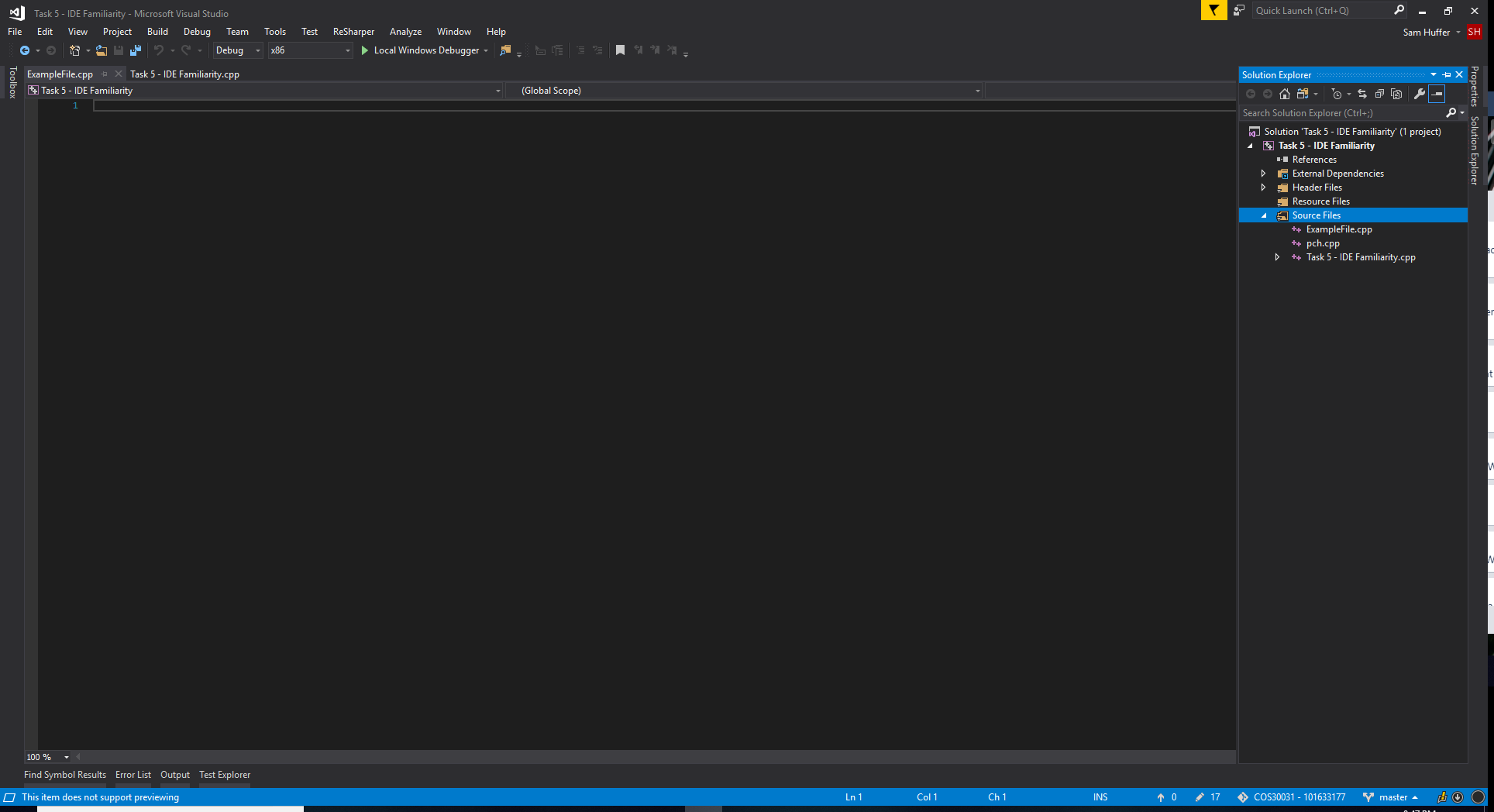
Once the project has been created, Visual Studio will open up a file editing tab for a C++ file conforming to its basic template. From here, you will be able to add, remove and change the code in that file at will.

Step 3: In this new dialog box, navigate to the folder you wish to store your project in, select it, and click “Select Folder”. If you navigate into the folder you wish to store your project in, click “Select Folder” without clicking on any sub-folders inside your chosen folder. Once this dialog box closes, leaving only the original, click “OK” in the bottom-right corner to create your project.

## 1.2. Adding new files

Step 1: Click on the docked “Solution Explorer” tab on the right-hand side of the screen (here seen under the “Properties” tab). This will open up the Solution Explorer tab. Clicking on “Source Files” to expand the folder will show you all the files you’ve coded that are currently in your project.

Step 2: To open a dialog box for adding a new file, right click anywhere on the Solution Explorer, and in the context menu that appears, click on “Add > New Item”.

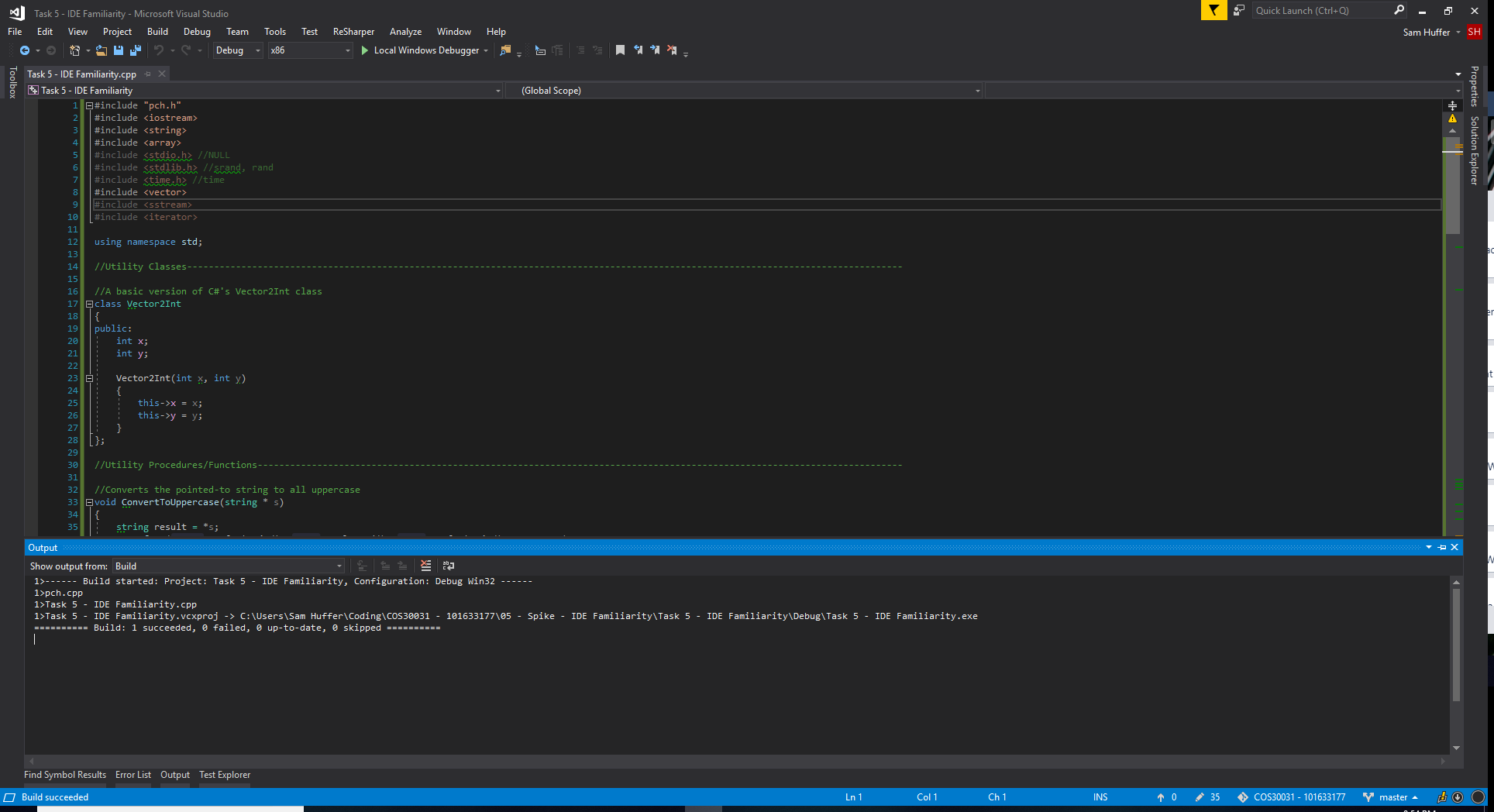


Once added, your new file will be opened in an editing tab next to any other files in the project you have open. This one will be completely blank. If you close it by clicking the “X” on the right of its tab, and wish to open it again later, it will now be available in the Solution Explorer in the “Source Files” folder.

Step 3: In the dialog box, select “C++ File (.cpp)” in the centre, and type in the name you wish to give your new file (e.g. “ExampleFile.cpp”). If you wish to place it within a particular sub-folder of your project, click “Browse” in the bottom-right corner to open up the dialog box you saw when you were choosing where to store your project in Step 3 of Creating a new command line project. Navigate to your desired folder and click “Select Folder”. Once complete, click “Add” in the bottom-right corner.

## 1.3. Compiling your program

Step 1: To build your program once you have some code you want to test, at the top of the screen, click “Build > Build Solution”. Shortcut keys: Ctrl + Shift + B.



If the build is successful, you will see the “Output” tab pop up down the bottom with output similar to this, saying “Build: 1 successful”.

## 1.4. Running the program

If there are bugs in your code, the build will fail and the “Error List” tab will pop up instead. It will outline what errors are in the code, and list which file they’re in and what line they’re on. Double click on an error to jump straight to its position in the code. Hint: click “Line” in the top-right corner of the “Error List” tab to order the errors by line number. Often an error at one point in the code will produce additional errors further down, even if the rest of the code is perfectly alright. For example, my incomplete assignment of a Vector2Int to the variable “pos” on line 221 (top-left of the screenshot) results in the compiler thinking that there are other errors below it. If I were to fix that error, any subsequent errors caused by it would be fixed. Note that unrelated errors would still remain and need to be fixed in their own right.

# 2. Debugging

## 2.1. Inserting breakpoints

## 2.2. Run program so it stops at the breakpoint

## 2.3. Inspecting values of variables during debugging