Visual Studio 2017 – QuickStart Guide

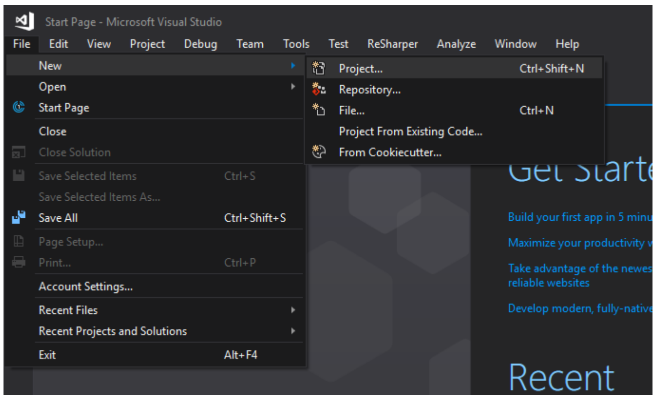
Version

This QuickStart guide was made for users of Visual Studio 2017 (Version 15.9.8, with JetBrains ReSharper installed. Due to ReSharper, some steps may vary from what will work for your version of Visual Studio).

Overview

1. Project Setup

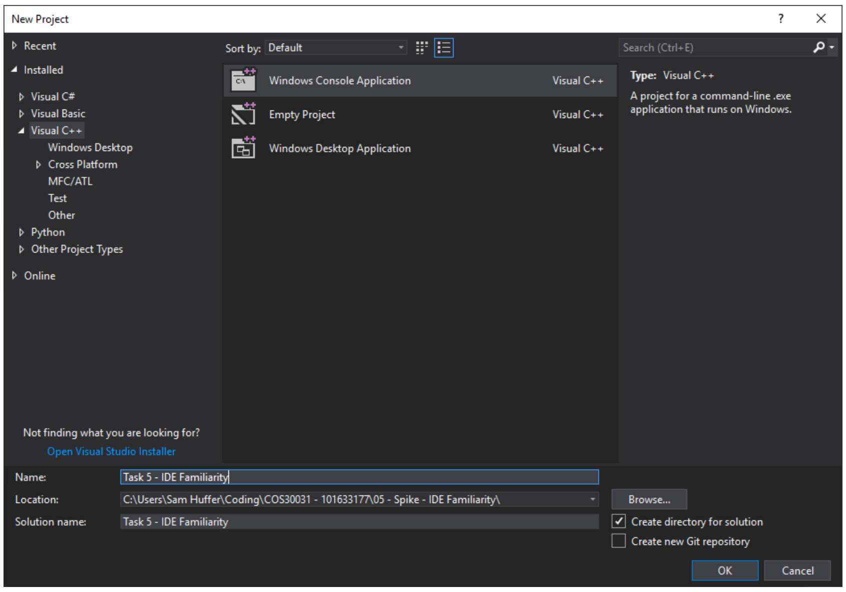
1.1. Creating a new command line project



Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test ReSharper Analyze Window Help
New
Open
Start Page
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project... Ctrl+Shift+N
Repository...
File... Ctrl+N
Project From Existing Code...
From Cookiecutter...
Get Starte
Build your first app in 5 minu
Maximize your productivity v
Take advantage of the newes reliable websites
Develop modern, fully-native
Recent

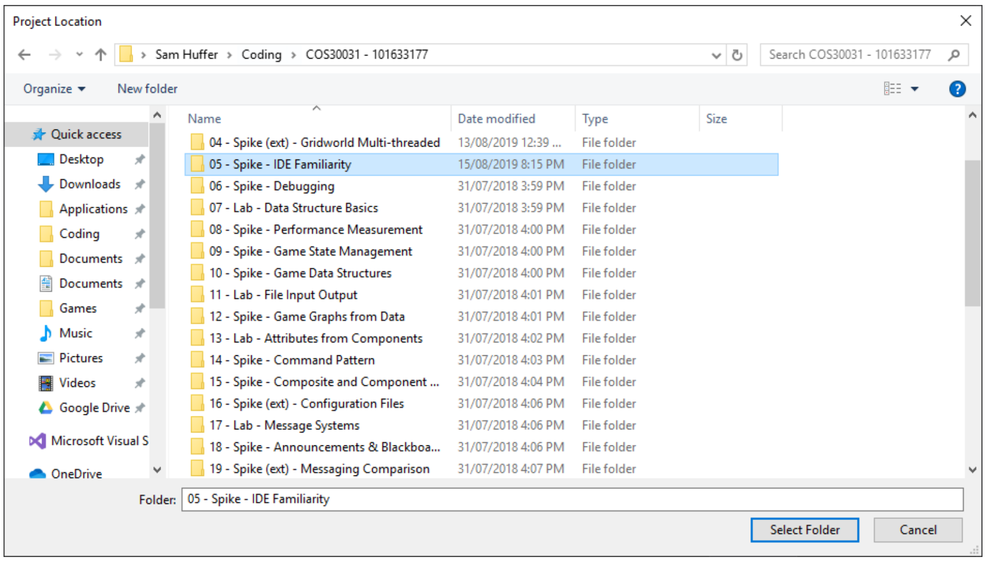
Step 1: In the top-left corner, click "File > New > Project". This will open a dialog box for creating a new project. Shortcut keys: Ctrl + Shift + N.

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.



Type: Visual C++
A project for a command-line .exe application that runs on Windows.

Name: Task 5 - IDE Familiarity
Location: C:\Users\Sam Huffer\Coding\COS30031 - 101633177\05 - Spike - IDE Familiarity\
Solution name: Task 5 - IDE Familiarity
☑ Create directory for solution
☐ Create new Git repository

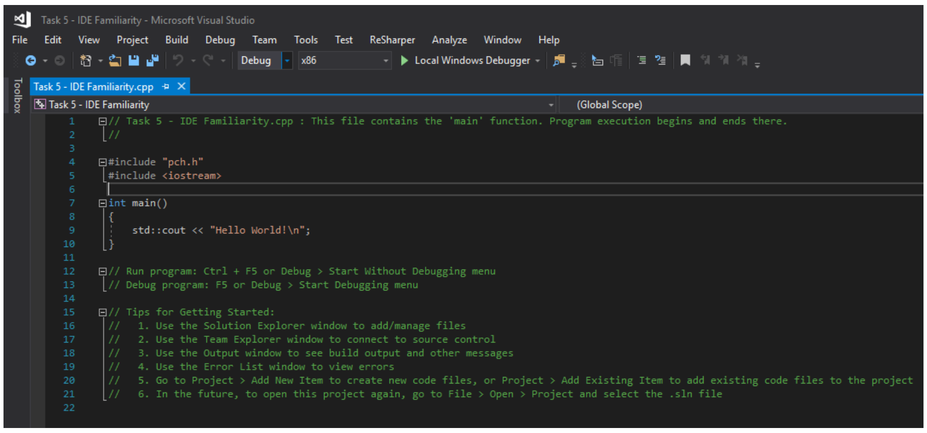


Name	Date modified	Type	Size
04 - Spike (ext) - Gridworld Multi-threaded	13/08/2019 12:39 ...	File folder	
05 - Spike - IDE Familiarity	15/08/2019 8:15 PM	File folder	
06 - Spike - Debugging	31/07/2018 3:59 PM	File folder	
07 - Lab - Data Structure Basics	31/07/2018 3:59 PM	File folder	
08 - Spike - Performance Measurement	31/07/2018 4:00 PM	File folder	
09 - Spike - Game State Management	31/07/2018 4:00 PM	File folder	
10 - Spike - Game Data Structures	31/07/2018 4:00 PM	File folder	
11 - Lab - File Input Output	31/07/2018 4:01 PM	File folder	
12 - Spike - Game Graphs from Data	31/07/2018 4:01 PM	File folder	
13 - Lab - Attributes from Components	31/07/2018 4:02 PM	File folder	
14 - Spike - Command Pattern	31/07/2018 4:03 PM	File folder	
15 - Spike - Composite and Component ...	31/07/2018 4:04 PM	File folder	
16 - Spike (ext) - Configuration Files	31/07/2018 4:06 PM	File folder	
17 - Lab - Message Systems	31/07/2018 4:06 PM	File folder	
18 - Spike - Announcements & Blackboa...	31/07/2018 4:06 PM	File folder	
19 - Spike (ext) - Messaging Comparison	31/07/2018 4:07 PM	File folder	

Folder: 05 - Spike - IDE Familiarity

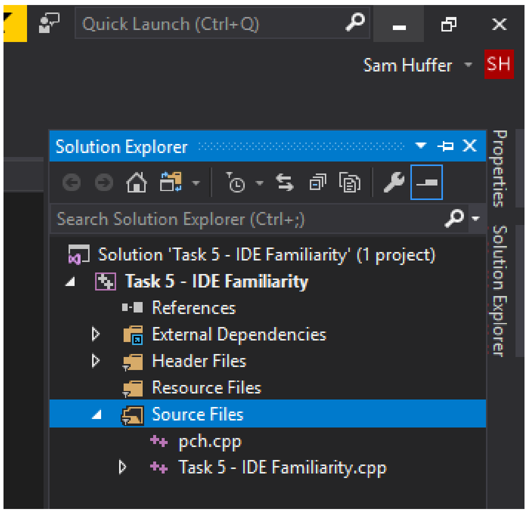
Step 3: In this new dialog box, navigate to the folder you wish to store your project in, select it, and click "Select Folder". If you navigate into the folder you wish to store your project in, click "Select Folder" without clicking on any sub-folders inside your chosen folder. Once this dialog box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

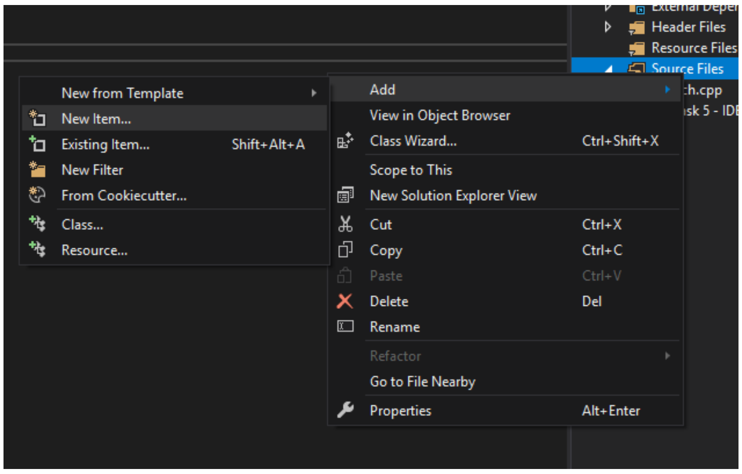
Once the project has been created, Visual Studio will open up a file editing tab for a C++ file conforming to its basic template. From here, you will be able to add, remove and change the code in that file at will.



```cpp
// Task 5 - IDE Familiarity.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include "pch.h"
#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
// Debug program: F5 or Debug > Start Debugging menu

// Tips for Getting Started:
//   1. Use the Solution Explorer window to add/manage files
//   2. Use the Team Explorer window to connect to source control
//   3. Use the Output window to see build output and other messages
//   4. Use the Error List window to view errors
//   5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project
//   6. In the future, to open this project again, go to File > Open > Project and select the .sln file
```

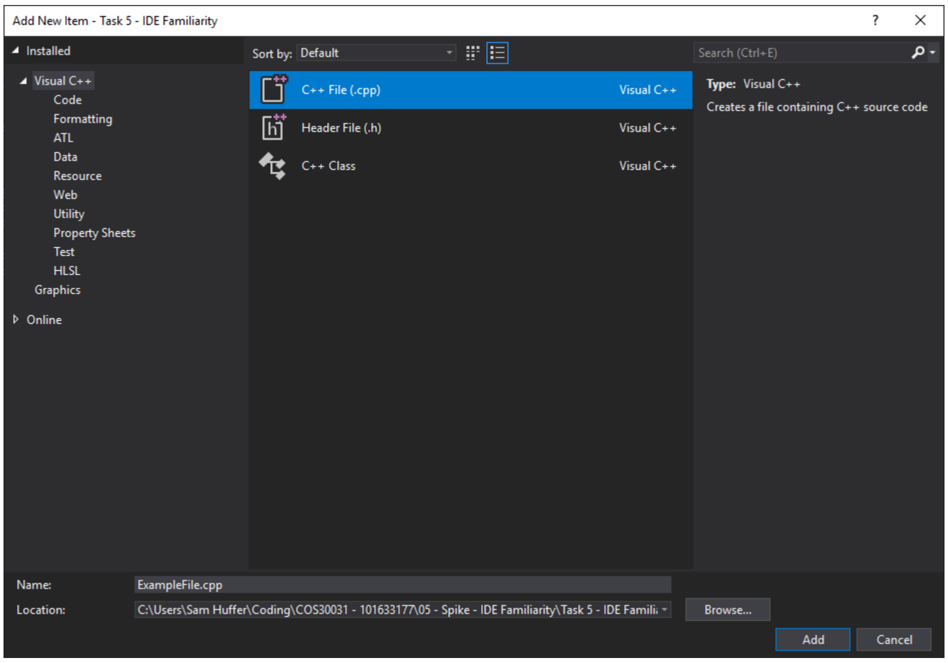
1.2. Adding new files





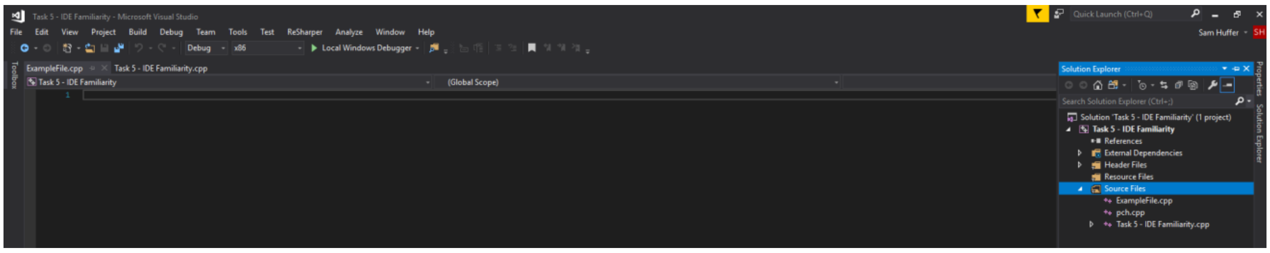
Step 1: Click on the docked "Solution Explorer" tab on the right-hand side of the screen (here seen under the "Properties" tab). This will open up the Solution Explorer tab. Clicking on "Source Files" to expand the folder will show you all the files you've coded that are currently in your project.

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".



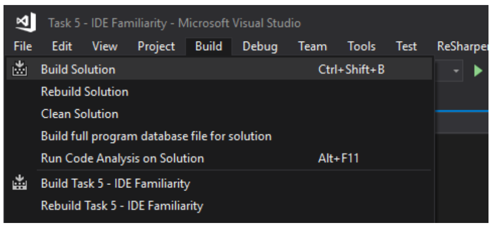
Name: ExampleFile.cpp
Location: C:\Users\Sam Huffer\Coding\COS30031 - 101633177\05 - Spike - IDE Familiarity\Task 5 - IDE Famili

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 Section 1.1, Step 3. Navigate to your desired folder and click "Select Folder". Once complete, click "Add" in the bottom-right corner.

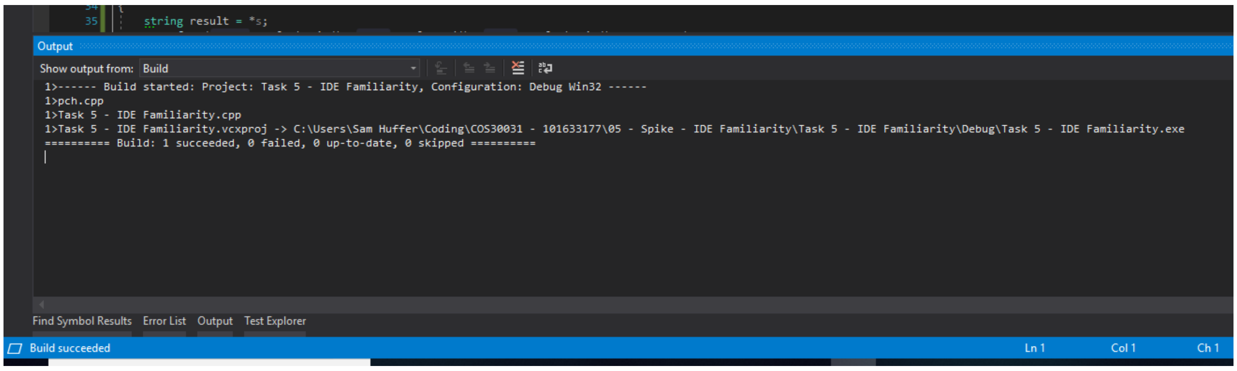

Task 5 - IDE Familiarity - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper Analyze Window Help
ExampleFile.cpp
Task 5 - IDE Familiarity.cpp
(Global Scope)
Solution Explorer
Solution 'Task 5 - IDE Familiarity' (1 project)
Task 5 - IDE Familiarity
References
External Dependencies
Header Files
Resource Files
Source Files
ExampleFile.cpp
pch.cpp
Task 5 - IDE Familiarity.cpp

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.

1.3. Compiling your program


Task 5 - IDE Familiarity - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Build full program database file for solution
Run Code Analysis on Solution Alt+F11
Build Task 5 - IDE Familiarity
Rebuild Task 5 - IDE Familiarity

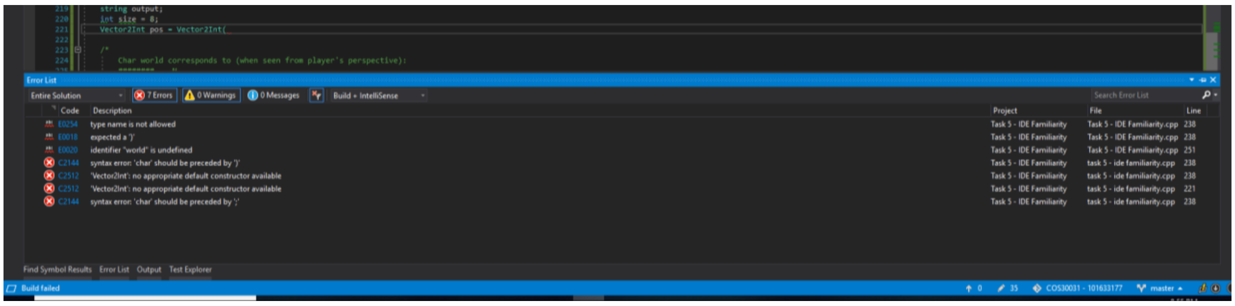
Step 1: To build your program once you have some code you want to test, at the top of the screen, click "Build > Build Solution". Shortcut keys: Ctrl + Shift + B.



```
string result = *s;
```

```
1>------ Build started: Project: Task 5 - IDE Familiarity, Configuration: Debug Win32 ------
1>pch.cpp
1>Task 5 - IDE Familiarity.cpp
1>Task 5 - IDE Familiarity.vcxproj -> C:\Users\Sam Huffer\Coding\COS30031 - 101633177\05 - Spike - IDE Familiarity\Task 5 - IDE Familiarity\Debug\Task 5 - IDE Familiarity.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

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".



```
219    string output;
220    int size = 8;
221    Vector2Int pos = Vector2Int(
222
223    /*
224        Char world corresponds to (when seen from player's perspective):
```

Error List

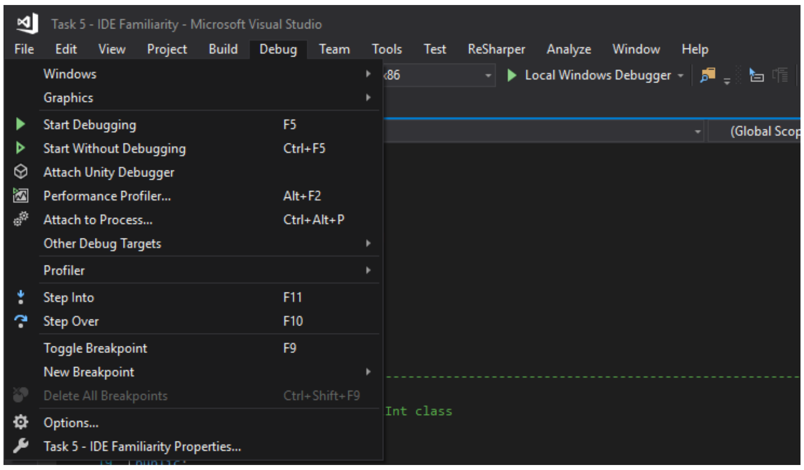
Entire Solution | 7 Errors | 0 Warnings | 0 Messages | Build + IntelliSense

	Code	Description	Project	File	Line
	E0254	type name is not allowed	Task 5 - IDE Familiarity	Task 5 - IDE Familiarity.cpp	238
	E0018	expected a ')'	Task 5 - IDE Familiarity	Task 5 - IDE Familiarity.cpp	238
	E0020	identifier "world" is undefined	Task 5 - IDE Familiarity	Task 5 - IDE Familiarity.cpp	251
	C2144	syntax error: 'char' should be preceded by ')'	Task 5 - IDE Familiarity	task 5 - ide familiarity.cpp	238
	C2512	'Vector2Int': no appropriate default constructor available	Task 5 - IDE Familiarity	task 5 - ide familiarity.cpp	238
	C2512	'Vector2Int': no appropriate default constructor available	Task 5 - IDE Familiarity	task 5 - ide familiarity.cpp	221
	C2144	syntax error: 'char' should be preceded by ';'	Task 5 - IDE Familiarity	task 5 - ide familiarity.cpp	238

Build failed

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.

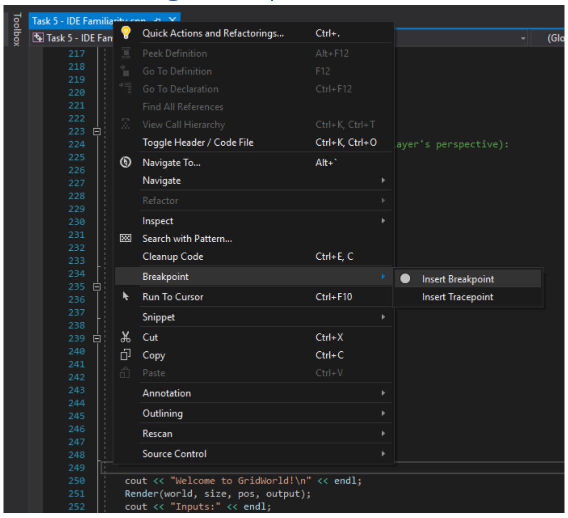
1.4. Running the program



Step 1: to run your program, you have several options. The easiest would be to click "Local Windows Debugger", which is in the top-centre of the screen next to a green "play" triangle. You can also click on "Debug > Start Debugging", which is equivalent to clicking "Local Windows Debugger", or "Debug > Start Without Debugging", which will also start your program but without running some of Visual Studio's debugging functionality. The shortcut keys for these options are F5 and Ctrl + F5 respectively. Note: each will also build your program before running it if you have saved any changes since you last built it.

2. Debugging

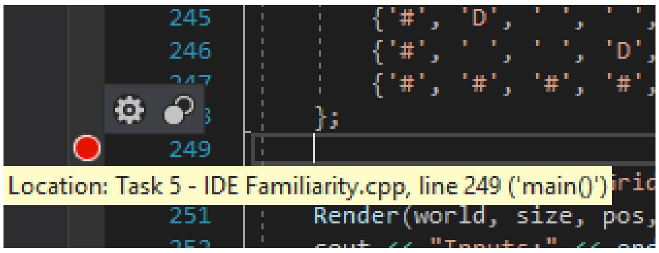
2.1. Inserting breakpoints



```
cout << "Welcome to GridWorld!\n" << endl;
Render(world, size, pos, output);
cout << "Inputs:" << endl;
```

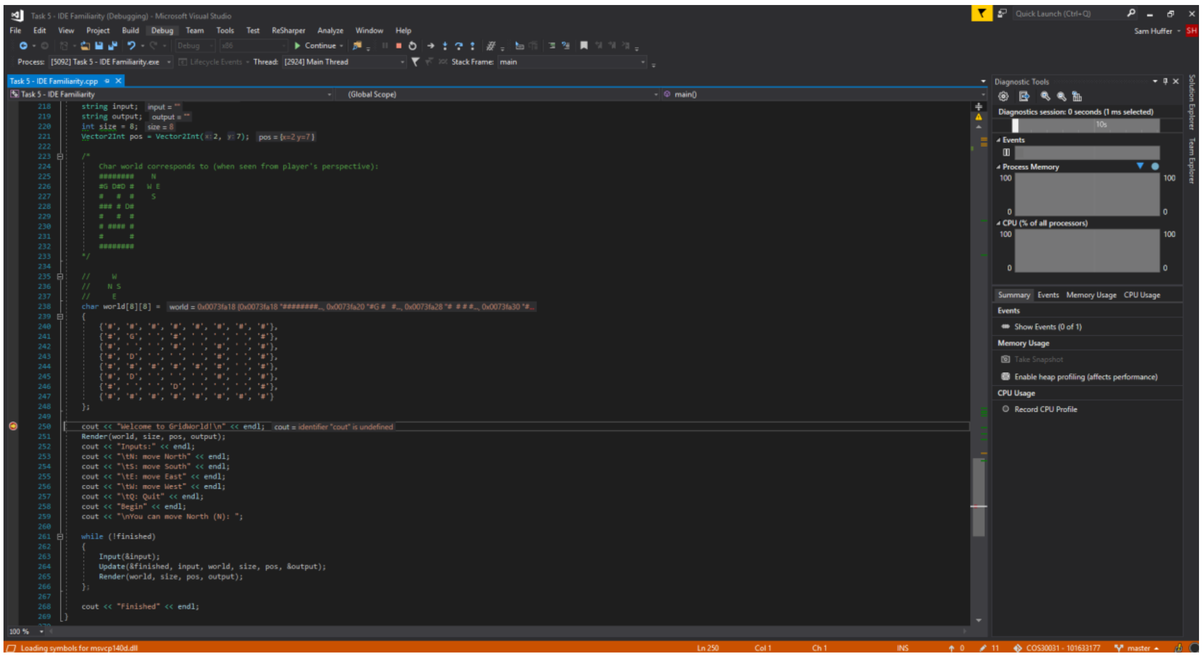
Step 1: to insert a breakpoint in your code such that it pauses execution at this point, right click on the line where you want to insert it, and in the context menu that appears, click on "Breakpoint > Insert Breakpoint".

This will insert a red dot beside the number of the line where it was inserted. To remove it, simply click on it. To disable it, hover your mouse over it, and a box with a cog and two circles will appear. Click on the two circles to enable and disable the breakpoint without removing it.



Location: Task 5 - IDE Familiarity.cpp, line 249 ('main()')

2.2. Run program so it stops at the breakpoint

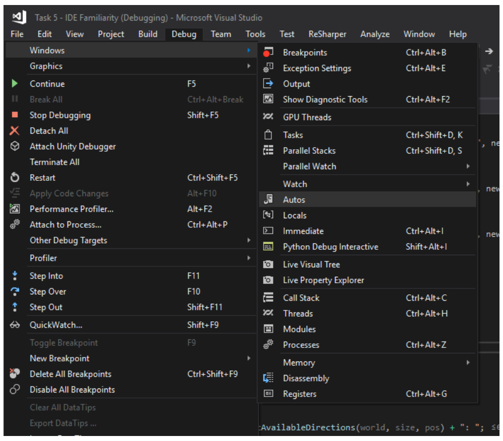


```
string input;  input = ""
string output;  output = ""
int size = 8;  size = 8
Vector2Int pos = Vector2Int(x: 2, y: 7);  pos = {x=2 y=7}

/*
    Char world corresponds to (when seen from player's perspective):
    ########    N
    #G D#D #   W E
    #   #  #    S
    ### # D#
    #   #  #
    # #### #
    #      #
    ########
*/

//     W
//    N S
//     E
char world[8][8] =  world = 0x0073fa18 {0x0073fa18 "########..., 0x0073fa20 "#G #  #..., 0x0073fa28 "#  # # #..., 0x0073fa30 "#...
{
    {'#', '#', '#', '#', '#', '#', '#', '#'},
    {'#', 'G', ' ', '#', ' ', ' ', ' ', '#'},
    {'#', ' ', ' ', '#', ' ', '#', ' ', '#'},
    {'#', 'D', ' ', ' ', ' ', '#', ' ', '#'},
    {'#', '#', '#', '#', '#', '#', ' ', '#'},
    {'#', 'D', ' ', ' ', ' ', '#', ' ', '#'},
    {'#', ' ', ' ', 'D', ' ', ' ', ' ', '#'},
    {'#', '#', '#', '#', '#', '#', '#', '#'}
};

cout << "Welcome to GridWorld!\n" << endl;  cout = identifier "cout" is undefined
Render(world, size, pos, output);
cout << "Inputs:" << endl;
cout << "\tN: move North" << endl;
cout << "\tS: move South" << endl;
cout << "\tE: move East" << endl;
cout << "\tW: move West" << endl;
cout << "\tQ: Quit" << endl;
cout << "Begin" << endl;
cout << "\nYou can move North (N): ";

while (!finished)
{
    Input(&input);
    Update(&finished, input, world, size, pos, &output);
    Render(world, size, pos, output);
};

cout << "Finished" << endl;
}
```

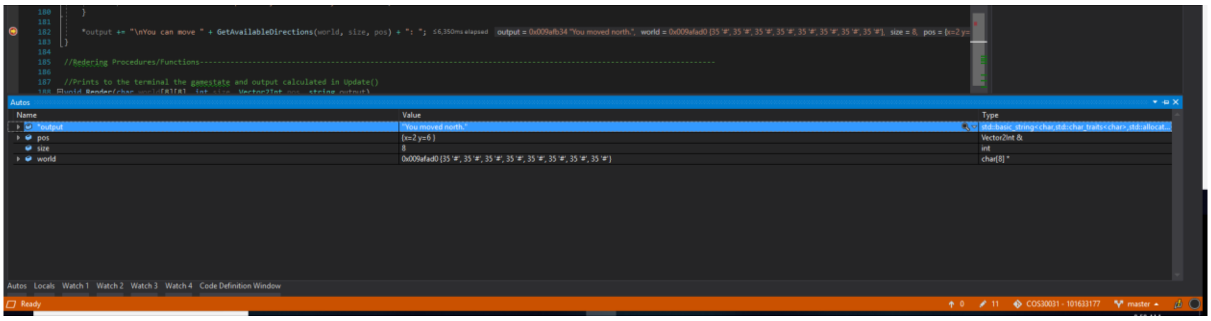
Task 5 - IDE Familiarity (Debugging) - Microsoft Visual Studio
Process: [5092] Task 5 - IDE Familiarity.exe
Thread: [2924] Main Thread
Stack Frame: main
Task 5 - IDE Familiarity.cpp
(Global Scope)
main()
Diagnostic Tools
Diagnostics session: 0 seconds (1 ms selected)
Events
Process Memory
CPU (% of all processors)
Summary
Events
Memory Usage
CPU Usage
Show Events (0 of 1)
Take Snapshot
Enable heap profiling (affects performance)
Record CPU Profile
Loading symbols for msvcp140d.dll
Ln 250
Col 1
Ch 1
INS
COS30031 - 101633177
master

Step 1: to run a program such that it stops at a breakpoint when that breakpoint is reached, click on "Local Windows Debugger" or "Debug > Start Debugging" as discussed in Section 1.4. The program will start, and when you reach a line with an enabled breakpoint, it will switch from the window the program is running in to the editor window at the line featuring the breakpoint, with the red dot of the breakpoint encapsulating a yellow arrow to indicate the line, in case you have multiple breakpoints. Note: do not click on "Debug > Start Without Debugging", as that will run the program without any debugging features, breakpoints included. If you run the program this way, Visual Studio will ignore your breakpoint.

2.3. Inspecting values of variables during debugging

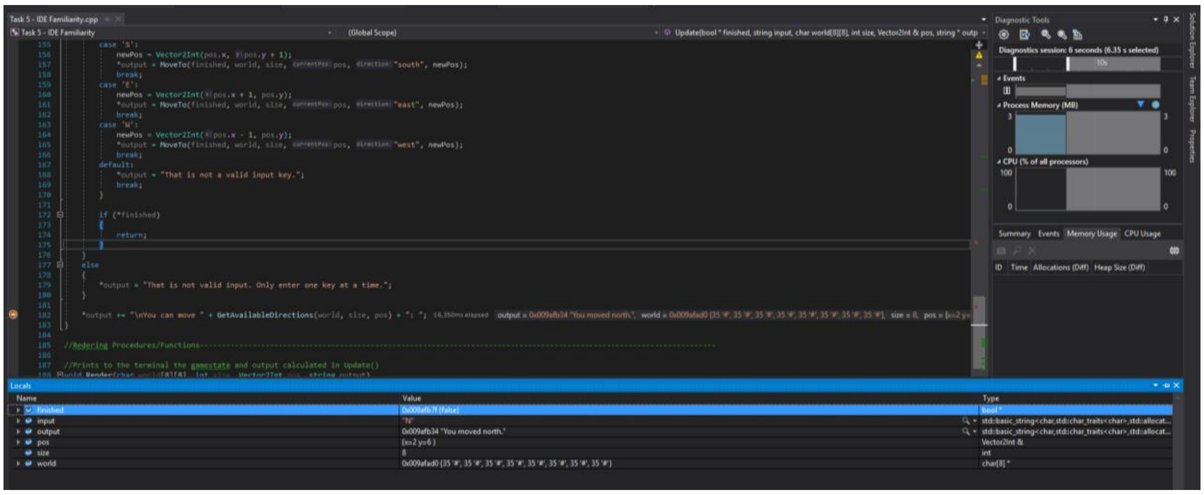


Once your program is running with debugging active, when it reaches a breakpoint, you may wish to inspect the values of variables in the code. There are several ways to do this. Firstly, to look at the values of variables present on the line of the breakpoint, click on "Debug > Windows > Autos" (left). This will open the Autos tab (below). (Once opened, this tab, if you don't click its "X" button to exit it, will be available in a tray of tabs at the bottom of the screen (below, bottom left).)



```
    }

    *output += "\nYou can move " + GetAvailableDirections(world, size, pos) + ": ";  ≤6,350ms elapsed  output = 0x009afb34 "You moved north.",  world = 0x009afad0 {35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#'},  size = 8,  pos = {x=2 y=
}

//Redering Procedures/Functions---------------------------------------------------------------------------------------------

//Prints to the terminal the gamestate and output calculated in Update()
void Render(char world[8][8], int size, Vector2Int pos, string output)
```

Name	Value	Type
*output	"You moved north."	std::basic_string<char,std::char_traits<char>,std::allocat...
pos	{x=2 y=6 }	Vector2Int &
size	8	int
world	0x009afad0 {35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#'}	char[8] *



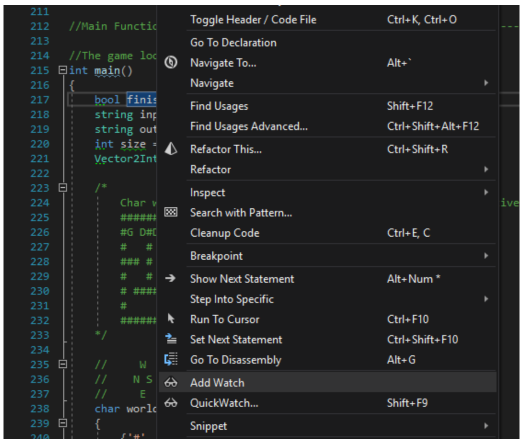
Task 5 - IDE Familiarity.cpp

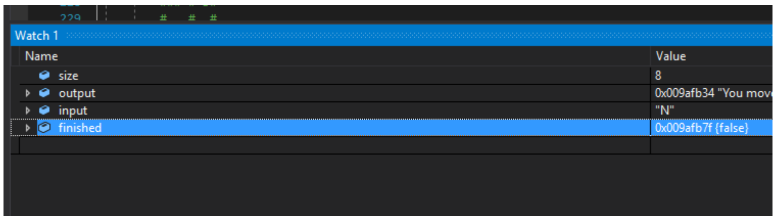
```
            case 'S':
                newPos = Vector2Int(pos.x, pos.y + 1);
                *output = MoveTo(finished, world, size, pos, "south", newPos);
                break;
            case 'E':
                newPos = Vector2Int(pos.x + 1, pos.y);
                *output = MoveTo(finished, world, size, pos, "east", newPos);
                break;
            case 'W':
                newPos = Vector2Int(pos.x - 1, pos.y);
                *output = MoveTo(finished, world, size, pos, "west", newPos);
                break;
            default:
                *output = "That is not a valid input key.";
                break;
            }

            if (*finished)
            {
                return;
            }
        }
        else
        {
            *output = "That is not valid input. Only enter one key at a time.";
        }

        *output += "\nYou can move " + GetAvailableDirections(world, size, pos) + ": ";
    }

    //Redering Procedures/Functions-----------------------------------------------

    //Prints to the terminal the gamestate and output calculated in Update()
```

Locals

Name	Value	Type
finished	0x009afb7f (false)	bool *
input	"N"	std::basic_string<char,std::char_traits<char>,std::allocat...
output	0x009afb34 "You moved north."	std::basic_string<char,std::char_traits<char>,std::allocat...
pos	{x=2 y=6 }	Vector2Int &
size	8	int
world	0x009afad0 {35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#', 35 '#'}	char[8] *

To view the values of all variables in the method currently being executed, you will instead want to click on "Debug > Windows > Locals". For example, my breakpoint above is in the Update method (you can see which method the line you're typing on is in at top-centre-right), so the Locals window is showing the values of all variables passed to and defined in Update.

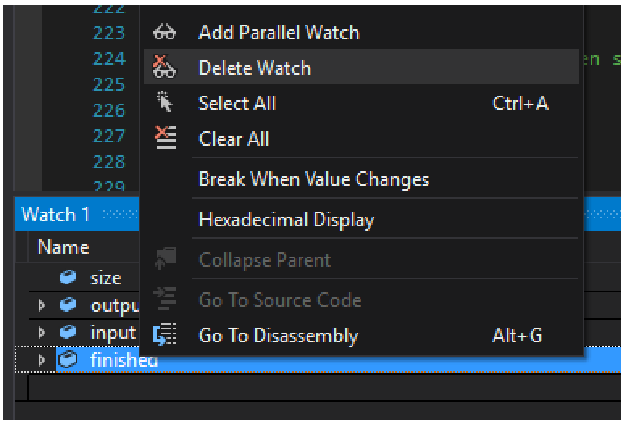




Watch 1

Name	Value
size	8
output	0x009afb34 "You mov
input	"N"
finished	0x009afb7f {false}

To view the values of specific variables throughout the execution of your program, regardless of where they are in the code compared to which line of code is being executed / paused on, right click on the variable in question and click "Add Watch" (above). This will open up the "Watch" tab and add the variable to the list of variables it is tracking (above-right). To stop tracking a variable, right click on it in the "Watch" tab and click "Delete Watch" (right).



Add Parallel Watch
Delete Watch
Select All Ctrl+A
Clear All
Break When Value Changes
Hexadecimal Display
Collapse Parent
Go To Source Code
Go To Disassembly Alt+G
Watch 1
Name
size
outpu
input
finished