Debugging Tools

Written by Rachel J. Morris, last updated Jan 22, 2018

About

Debugging is an important part of programming. It can be difficult enough to debug *your own* code, but in the professional world you will end up debugging lots and lots and lots of *other peoples' code*.

Debugging tools are a must.

Topics

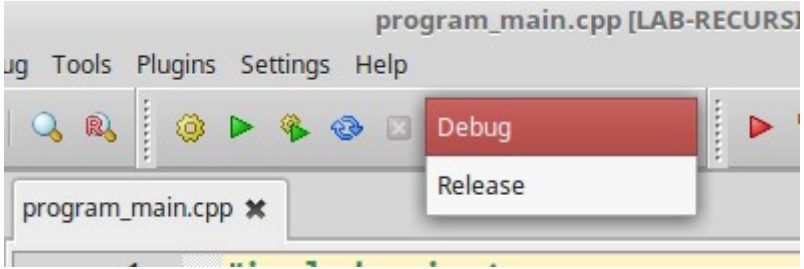
1. Running in Debug Mode
2. Breakpoints & Stepping
3. Watching variables
4. Call stack
5. Extra – Navigating code tips

Running in Debug Mode

1. Running in Debug Mode

When you're working on a program, you should always be running it in <u>debug mode</u>.

Once you're ready to release your program, then you make a <u>release</u> build.



The Debug/Release dropdown menu in Code::Blocks; it is similar in Visual Studio.

Notes

1. Running in Debug Mode

When you run your program in debug mode, it allows you to use debugging tools.

Additionally, when your program crashes, it will give you a chance to <u>pause</u> the program execution and investigate the problem.

Segfault in Code::Blocks

Notes

1. Running in Debug Mode

program_main.cpp

```
8   #include "function4.hpp"
9   #include "function5.hpp"
10
11  void ClearScreen();
12  void Pause();
13
14  /*** DO NOT MODIFY MAIN *******
15
16  int main()
17  {
18      int * blah = nullptr;
19      cout << *blah;
20
21
```

Watches (new)

Function arguments	
Locals	
blah	0x0
quit	false

Call stack

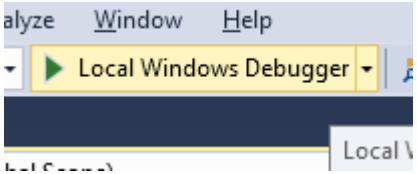
Nr	Address	Function	File
0	0x40157c	main()	/home/rayechell/TEACHING/

Code::Blocks pausing to investigate bad code.

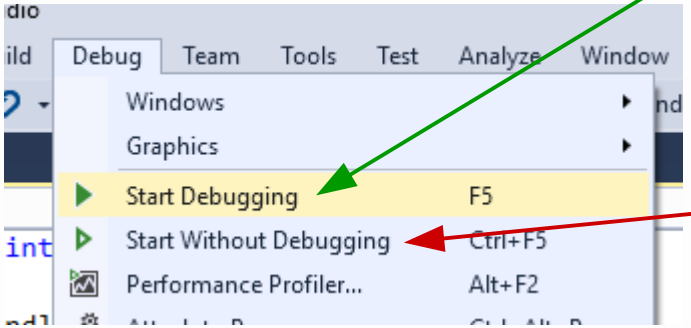
Notes

1. Running in Debug Mode

So make sure to always run your programs in DEBUG MODE!



Click me!



NOT ME!!

Notes

Breakpoints & Stepping

2. Breakpoints & Stepping

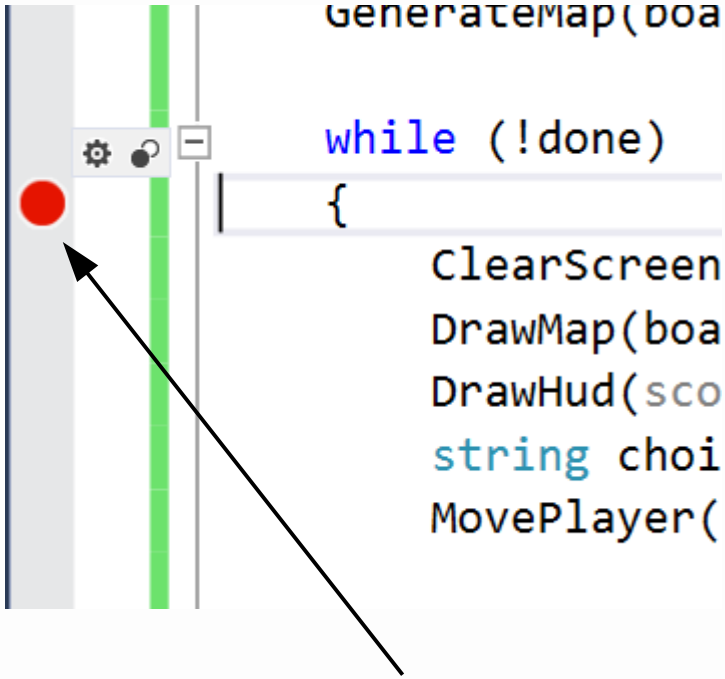
When you have a logic error in your program, it might be hard to find the error. Logic errors don't always cause crashes (such as just a bad formula), so you'll really have to investigate to find the problem.

That's where **Breakpoints** come in.

Notes

2. Breakpoints & Stepping

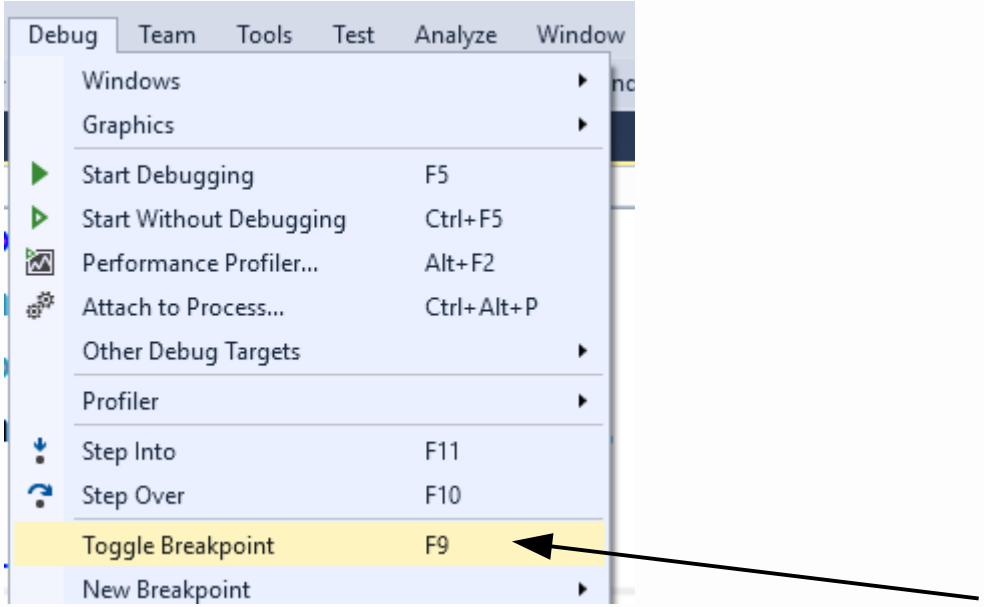
To set a breakpoint in your code, you can click in the gray bar to the left of the code window



Notes

2. Breakpoints & Stepping

Or you can choose it from the dropdown menu when you have a line of code selected.



Notes

2. Breakpoints & Stepping

When a breakpoint is hit in debug mode the program will pause execution and take you to the IDE.

From a paused state, you can step through the code execution line-by-line, as well as investigate variable values.

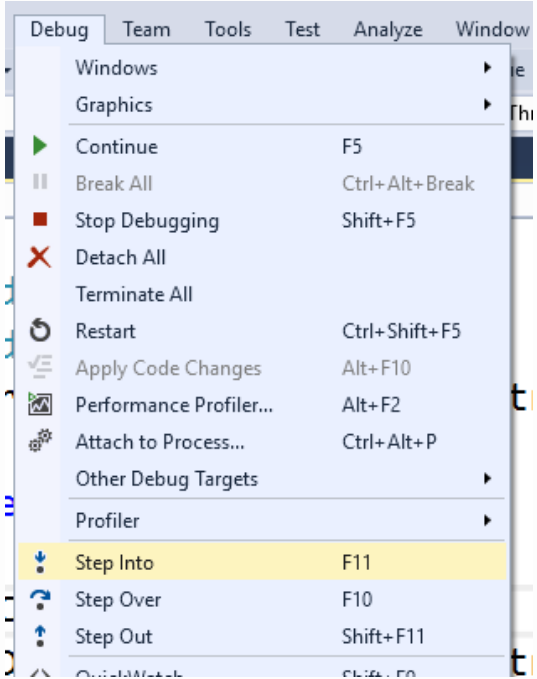
It paused at this line.

```
while (!done)
{
    ClearScreen();
    DrawMap(board, player, tre
    DrawHud(score);
```

Notes

2. Breakpoints & Stepping

There are toolbar options to step through the code



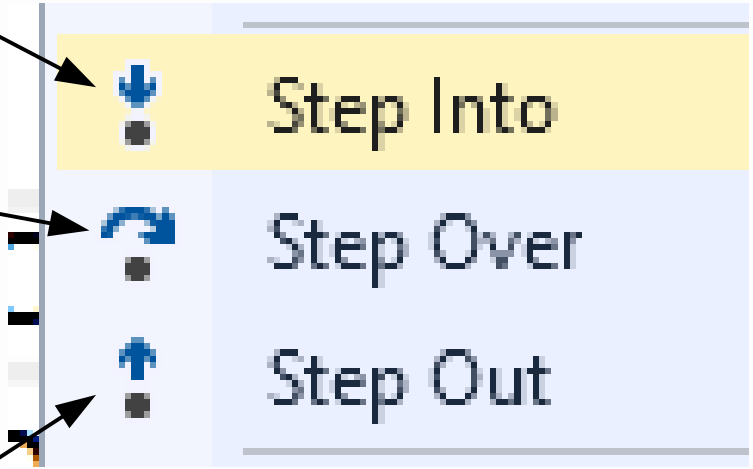
Or you can access the options from the Debug drop-down menu.

Notes

2. Breakpoints & Stepping

When at a function, step *into* the function's execution.

When at a function, step *over* it and go to the next line at the current scope



When inside a function, finish execution and step to the line after that function's call.

Notes

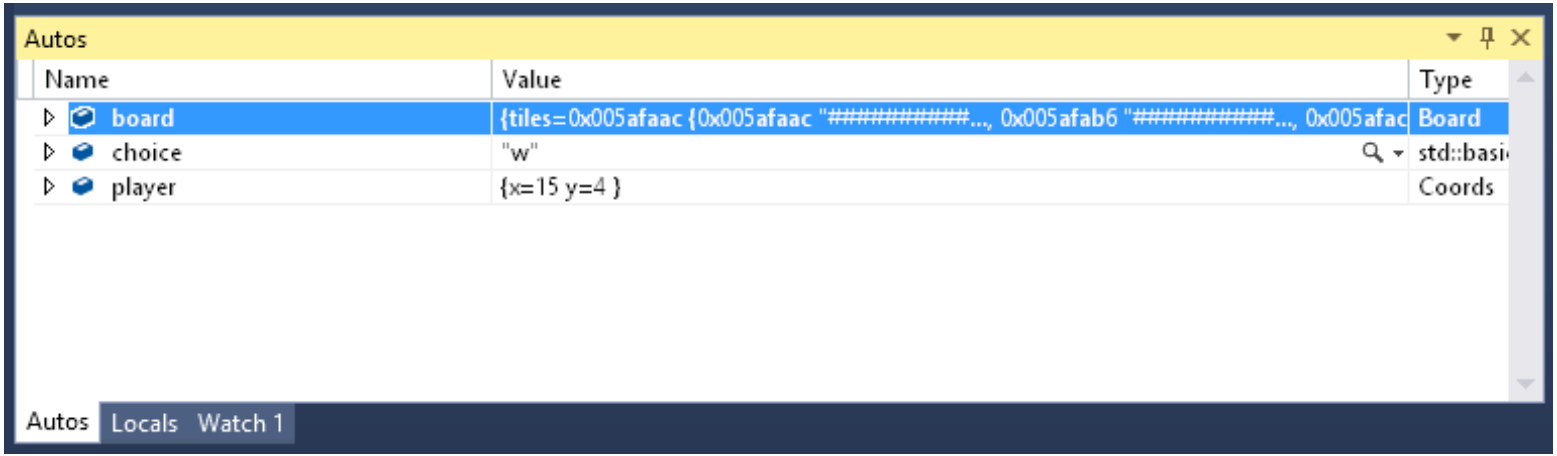
Watching Variables

3. Watching Variables

While you're paused, you will probably want to check in on the value of some variables. There are several windows where you can do this.

Notes

3. Watching Variables

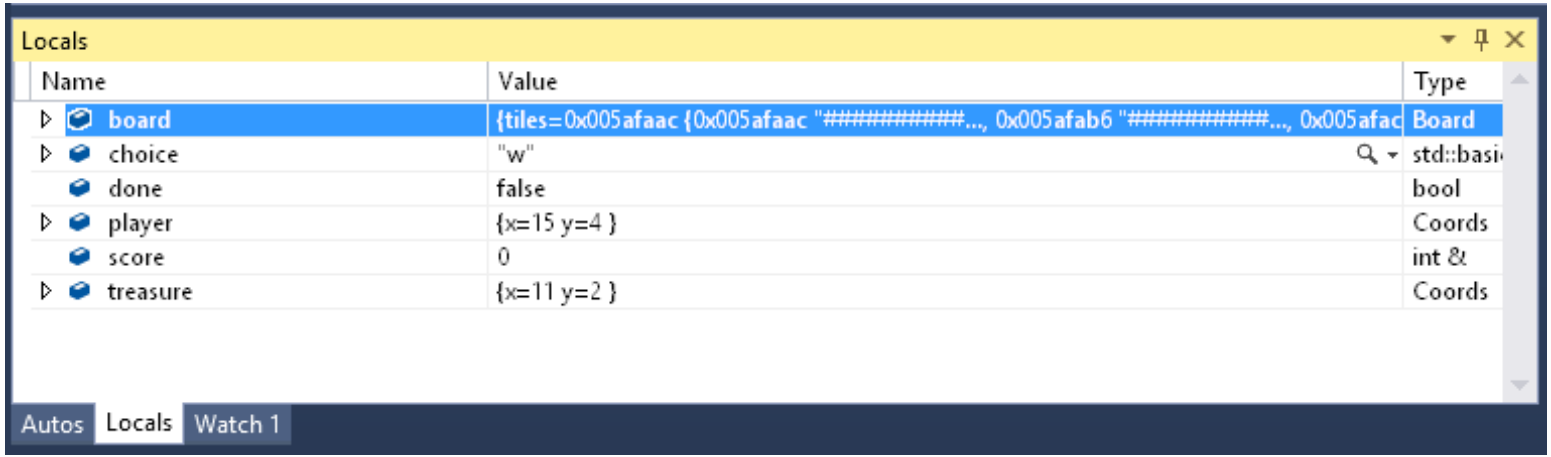


The Autos pane will show you any variables used on the current or preceding line of code.

With structs/classes and arrays, you can use the ► icon to show member variables or elements & their values.

Notes

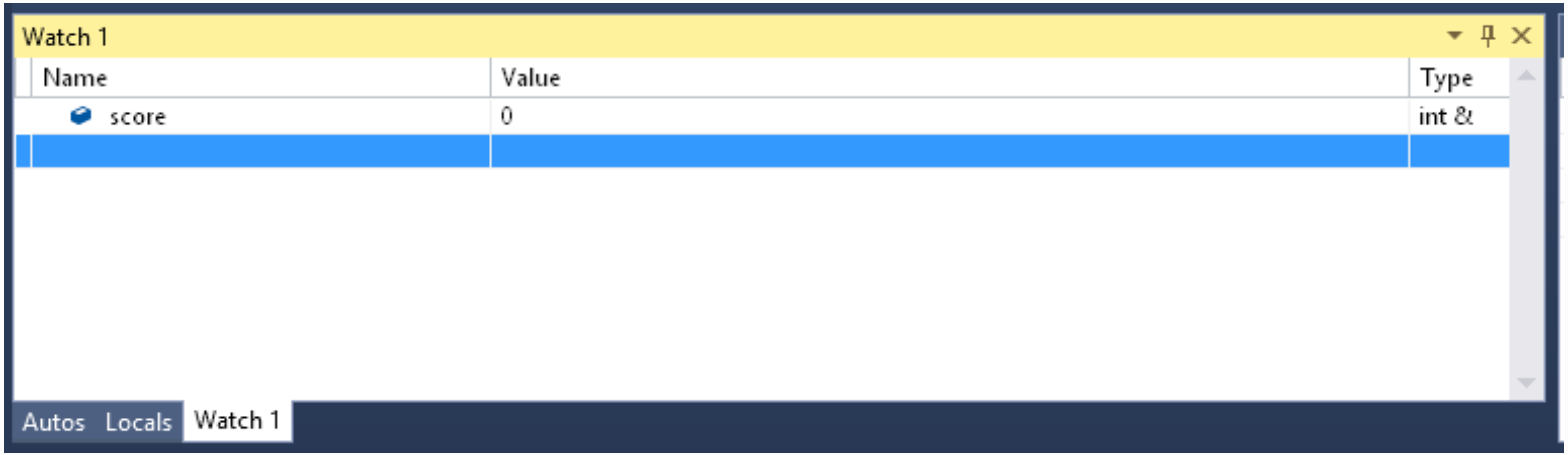
3. Watching Variables



The Locals pane will show you variables that are currently in scope.

Notes

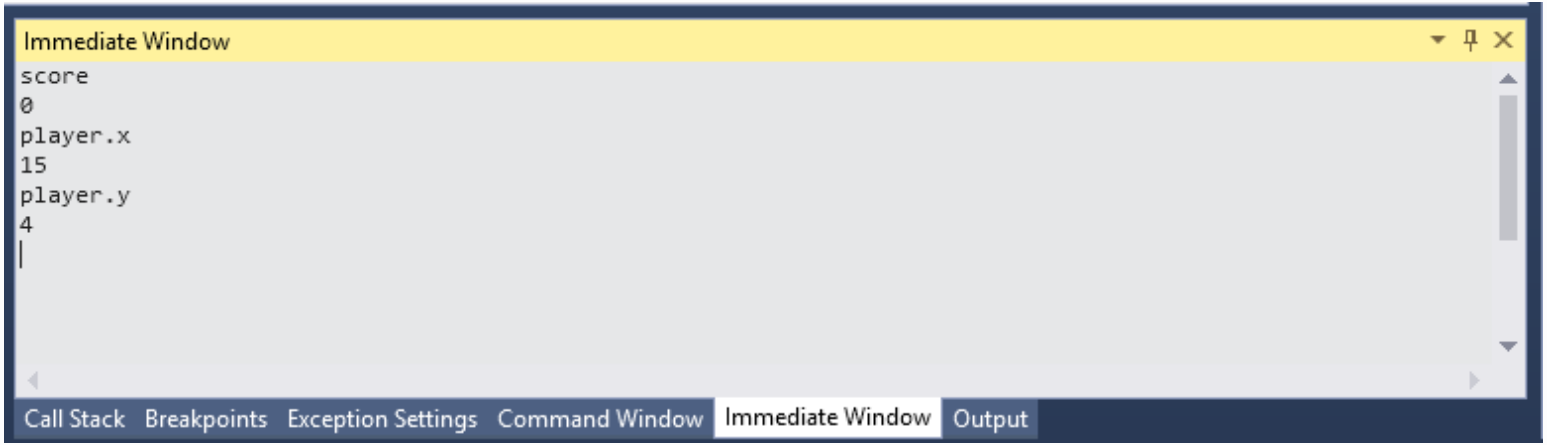
3. Watching Variables



In the Watch pane, you can type in a variable name to track it throughout the entire program execution.

Notes

3. Watching Variables



And if you want to see a variable value quickly, you can use the Immediate Window to type in a variable name. After you hit ENTER it will show you that variable's current value.

Notes

Call Stack

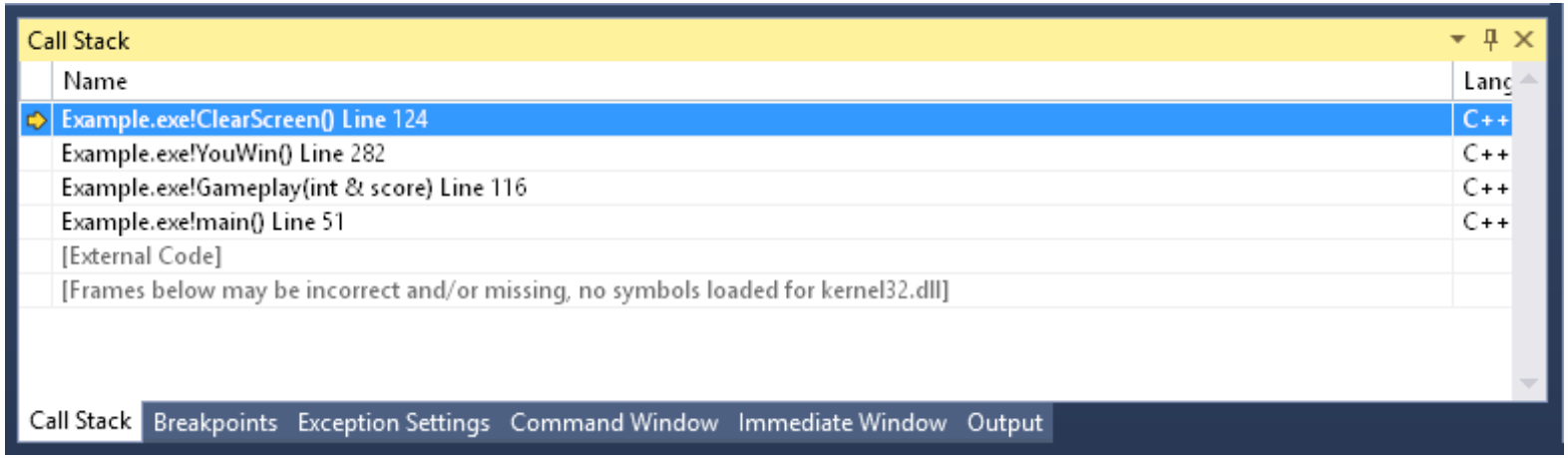
4. Call Stack

Another useful debugging tool is the Call Stack, though it might look intimidating at first.

When your program execution is paused, the Call Stack will show you the functions that have been called in order to get to where you're currently at.

Notes

4. Call Stack



The Call Stack can be invaluable to trace the path that your program is following.

Above, we can read that:

- main() called Gameplay(int& score),
- Gameplay() called YouWin(),
- YouWin() called ClearScreen()

Notes

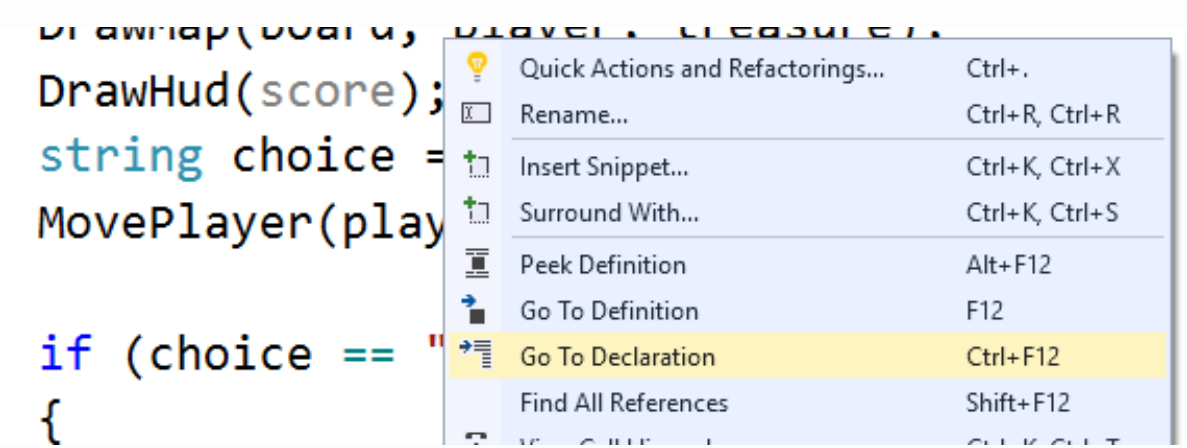
Extra: Tips for navigating code

5. Extra Tips

Here are some extra tips for navigating your code…

Notes

5. Extra Tips

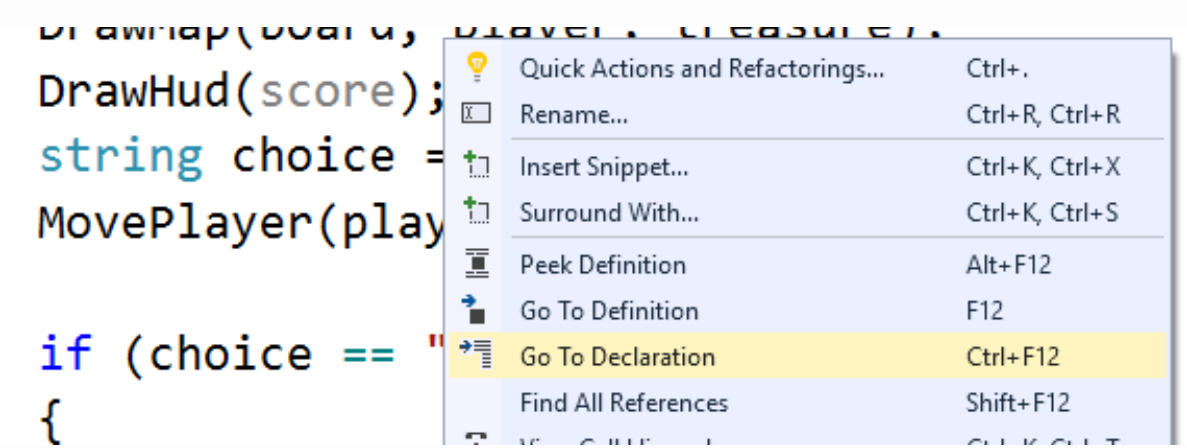


When you right-click on a function, you can choose "Go To Declaration" to go to the function declaration.

If it's a class method, it will go to the .h/.hpp file where the function is declared.

Notes

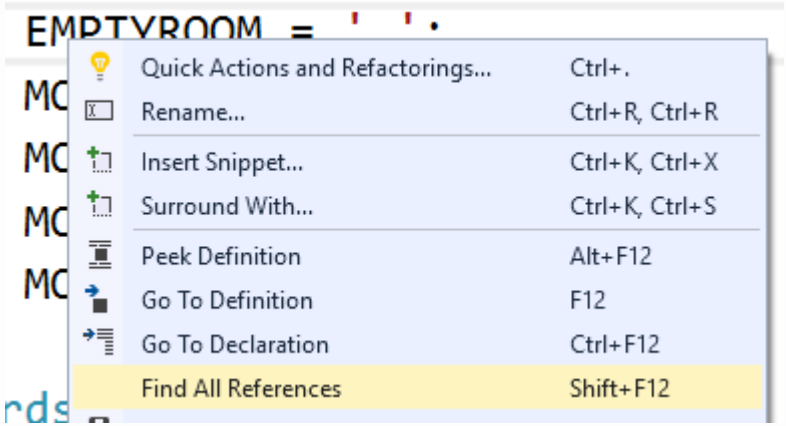
5. Extra Tips



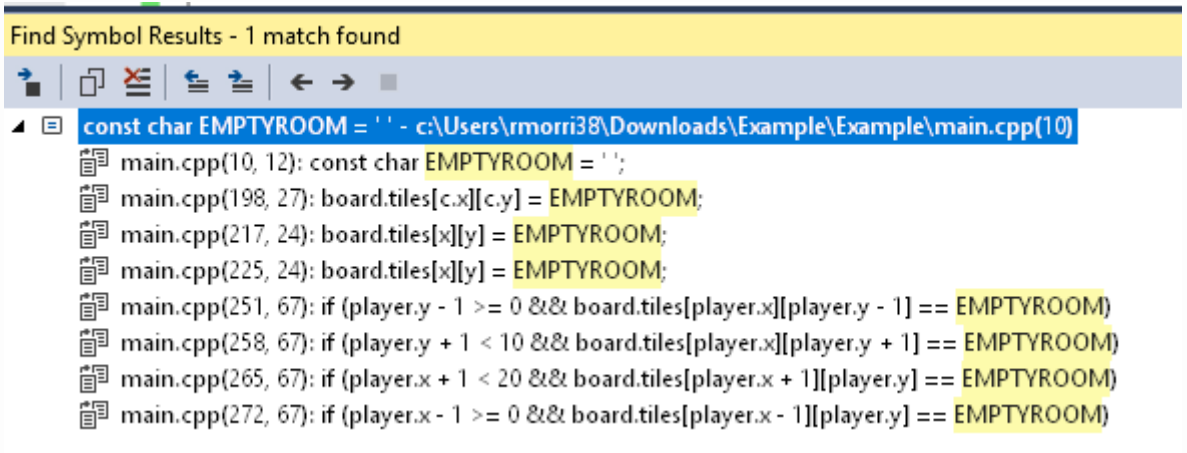
The “Go To Definition” option will go to the implementation of the function. If it is a class method, this is in the .cpp file.

Notes

5. Extra Tips

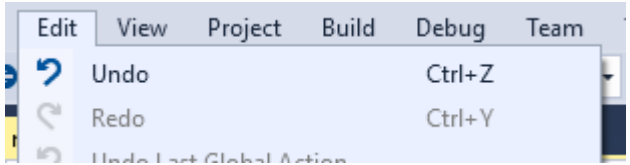


The “Find All References” will show you where in the code some variable is being used.

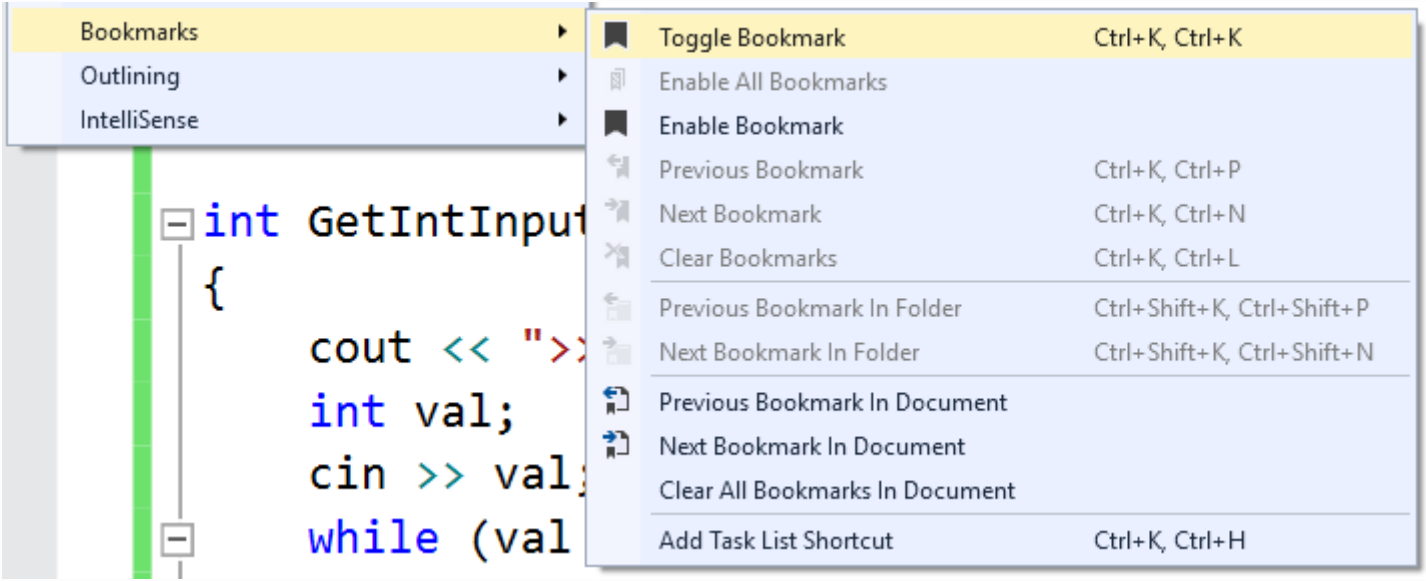


Notes

5. Extra Tips



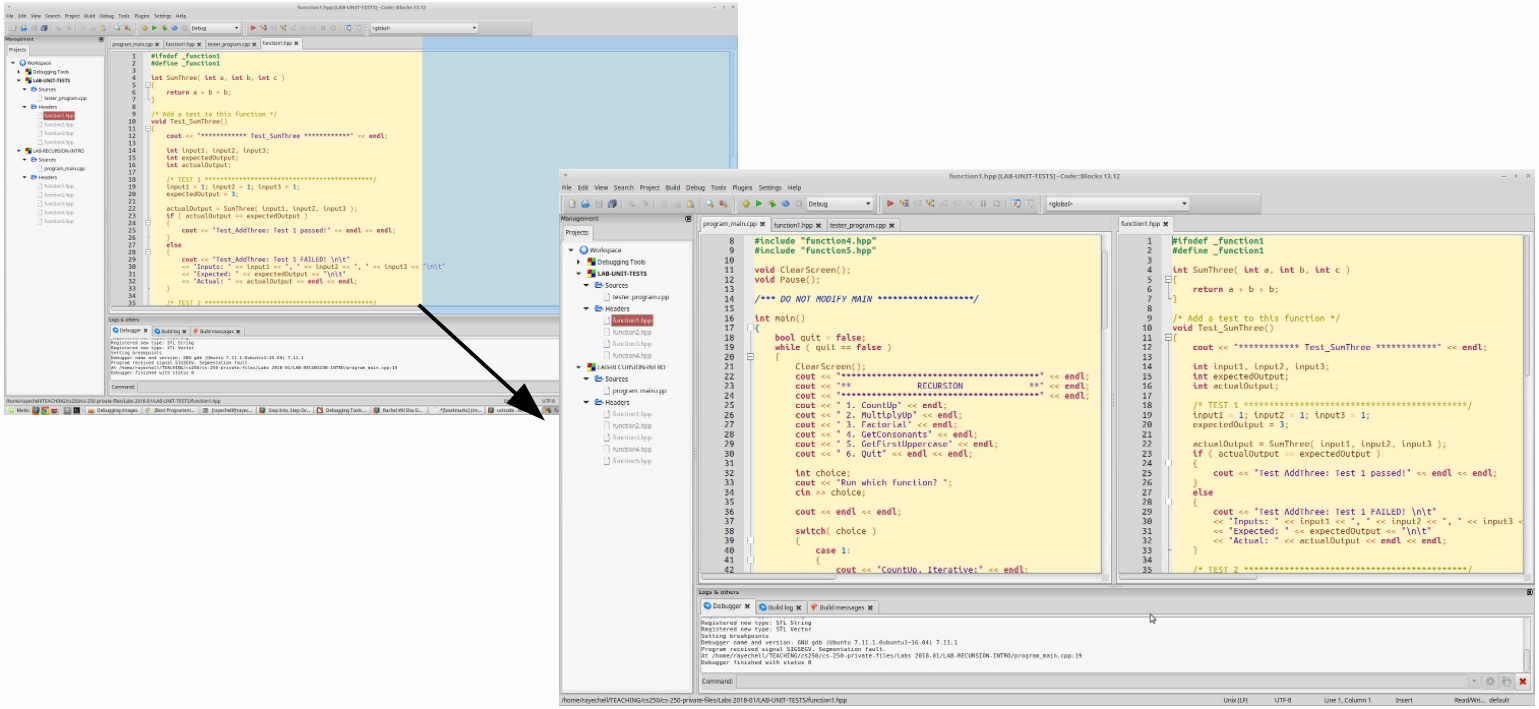
In the Edit > Bookmarks menu, you can set places in your code to access quickly.



Notes

5. Extra Tips

If you click and drag one file to the side of another, Visual Studio and Code::Blocks will let you show both windows side-by-side.



Notes

5. Extra Tips

You can also split a single code file view under Window > Split so that you can reference two areas in the same code file at once.

Notes

Conclusion

This was just a quick overview of debugging tools that you can (should) use.

The more you use them, the better you will get at debugging and the more independent of a developer you will be.