**Part 2: Analysis - 1:**

**Question:**

**Learn to use your debugger, this can be done before separating into individual files. Note that all enrolled students have access to Visual Studio 2019 through SMC Virtual Lab:https://www.smc.edu/administration/information-technology/vcl/**

**Place a breakpoint in the function Simulation::one(int t) at the line int k = watchme; In the original provided source it will be line 457. From there you should be able to identify the values for the member variables for the Simulation object. In particular, note that the starting value for Simulation:: watchme is 1. Step through the rest of the code and make note each time Simulation::watchme changes in value. Do this until you find yourself in the main function. In**

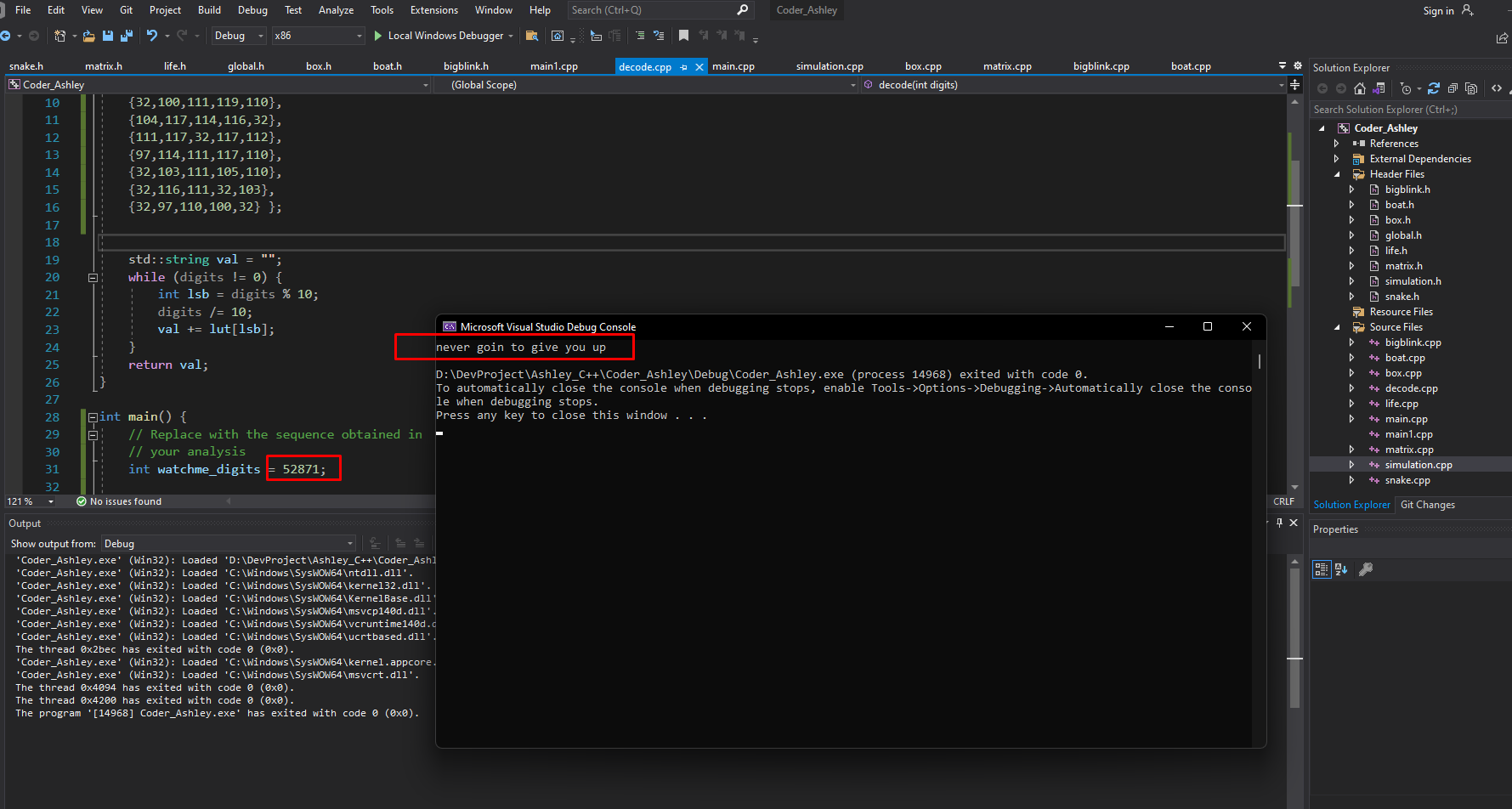
**VS2019 this occurs in about 40 steps. You should find that Simulation::watchme changed four times; so it took on total of five different values including the starting 1. These values are part of a five digit number with the first, 1, being the least significant digit. Once you have identified the entire five digit number, pass it into the decode function provided in the short program provided in decode.cpp(create a separate project to run this). What is the decoded sequence of characters? If the decoded sequence does not make sense, then you may have entered the sequence into the decoder in the incorrect order. There should be no duplicate values.**

**Answer :**

***Short Answser:***

The value of “watchme” varible is changed from 1 to 7, 8, 2, 5.

So the identified entire five digit number is “52871” and the decoded sequence of characters was “never goin to give you up”.

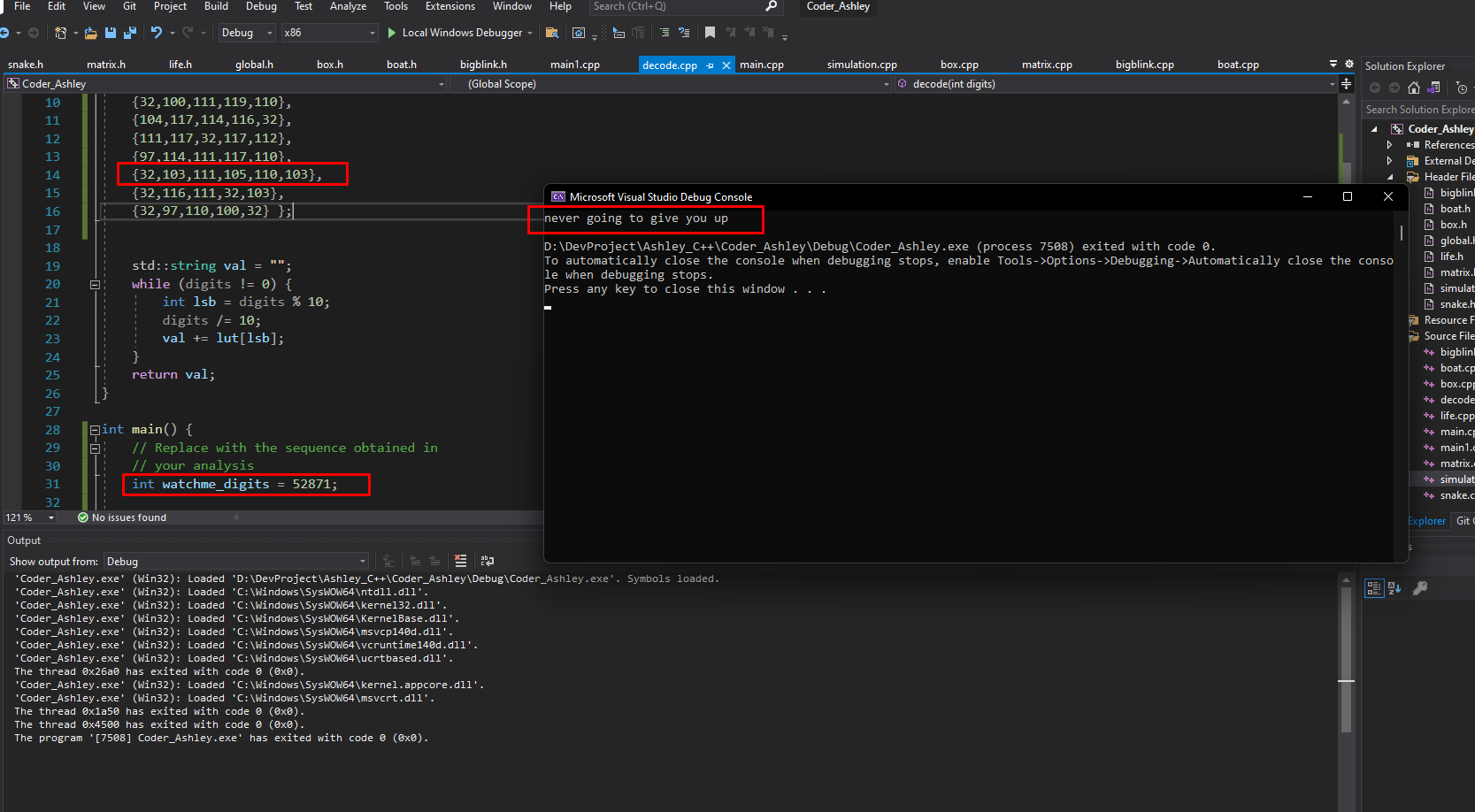
****

**Figure. Screenshot after running the original decoder.cpp**

**%%%Bonus**

Since this is not an accurate English sentence, I modified the value of the 7th element of the array from {32,103,111,105,110}(" goin") to {32,103,111,105,110,103}(" going").

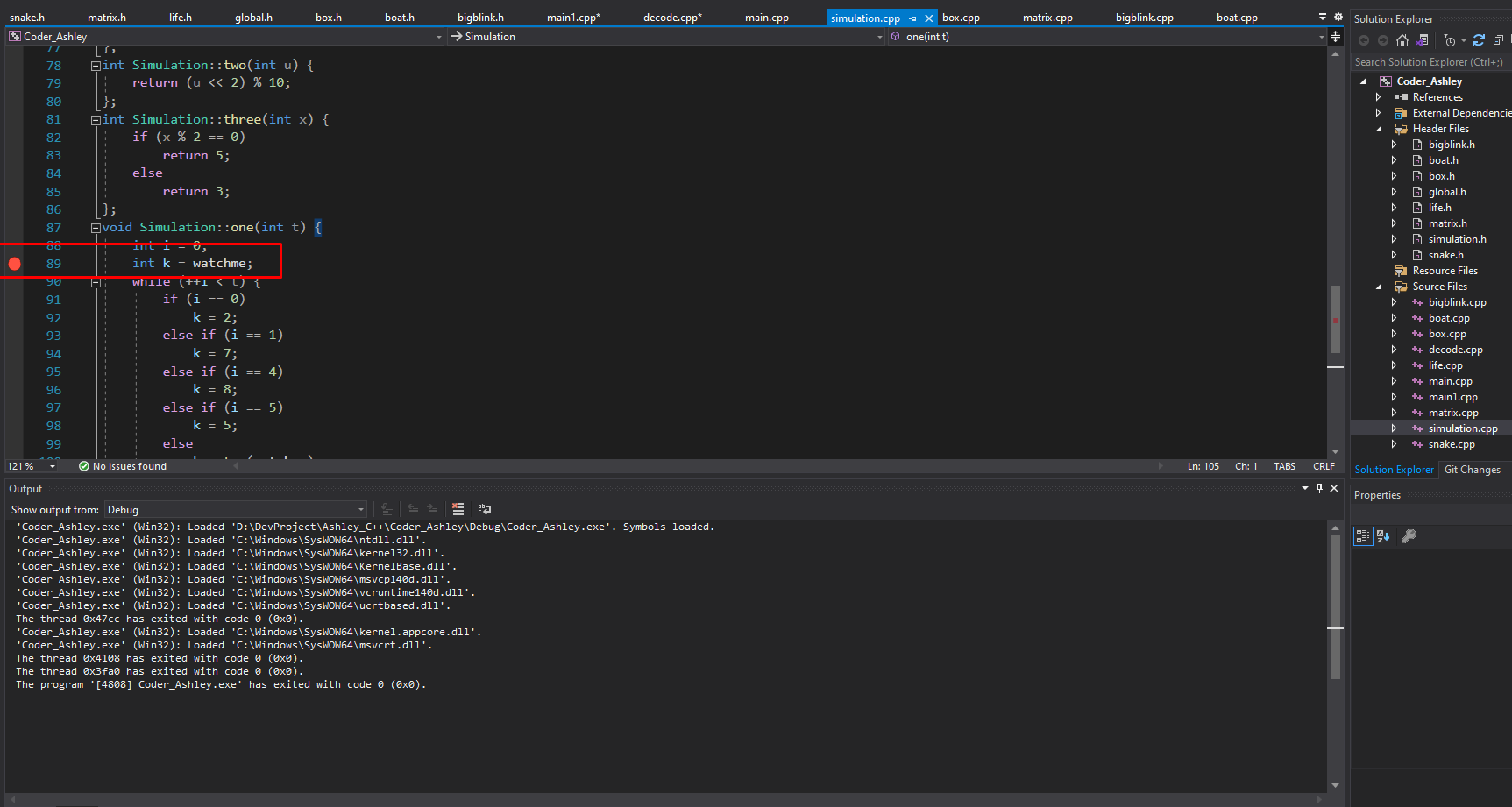
After modification, the decoded sequence of characters was “never going to give you up”.

****

**Figure. Screenshot after running the modified decoder.cpp**

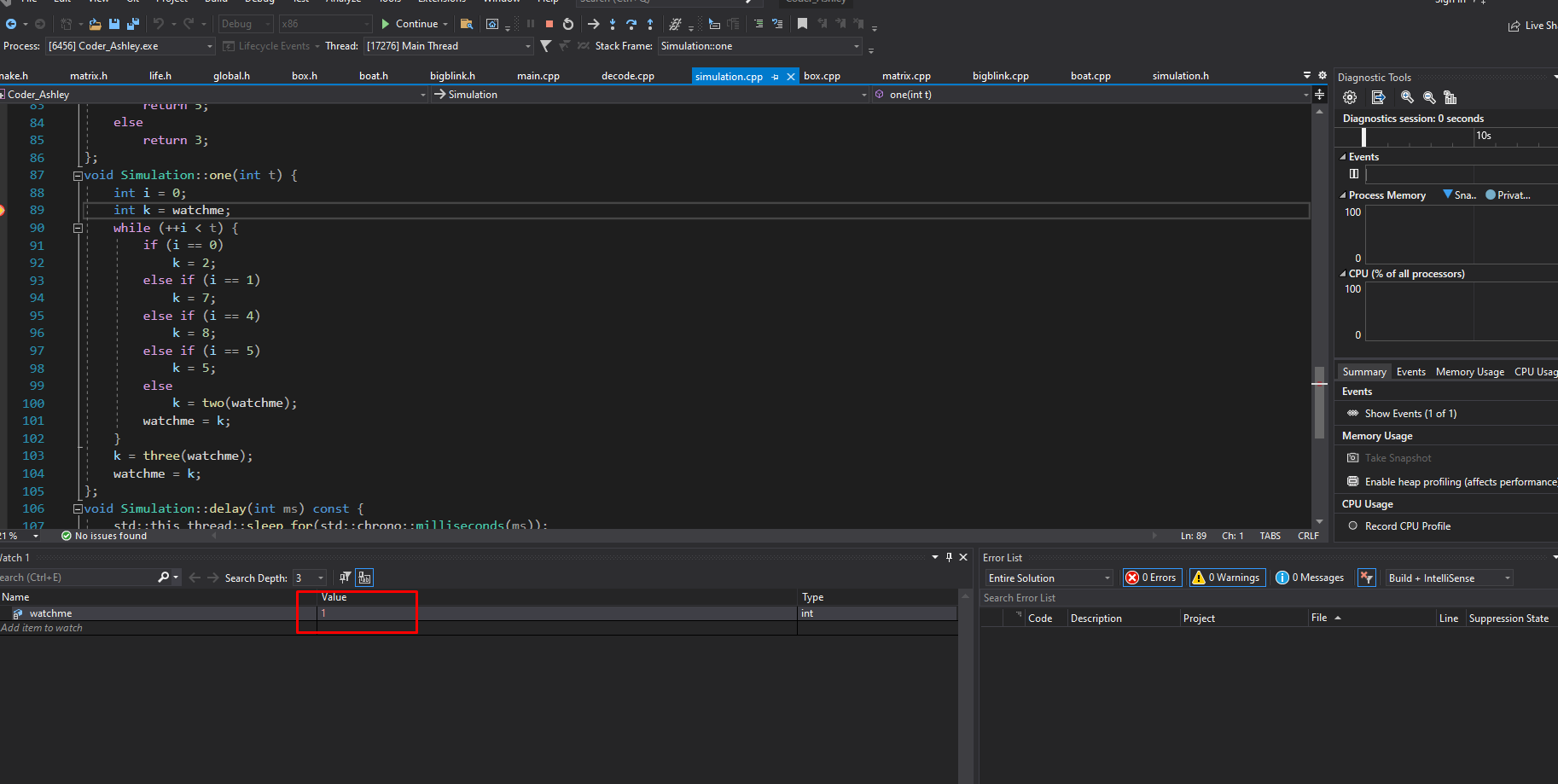
***Solution Process:***

* The value of “watchme” varible is changed from 1 to 7, 8, 2, 5.
* Place a breakpoint in the function Simulation::one(int t) at the line int k = watchme;

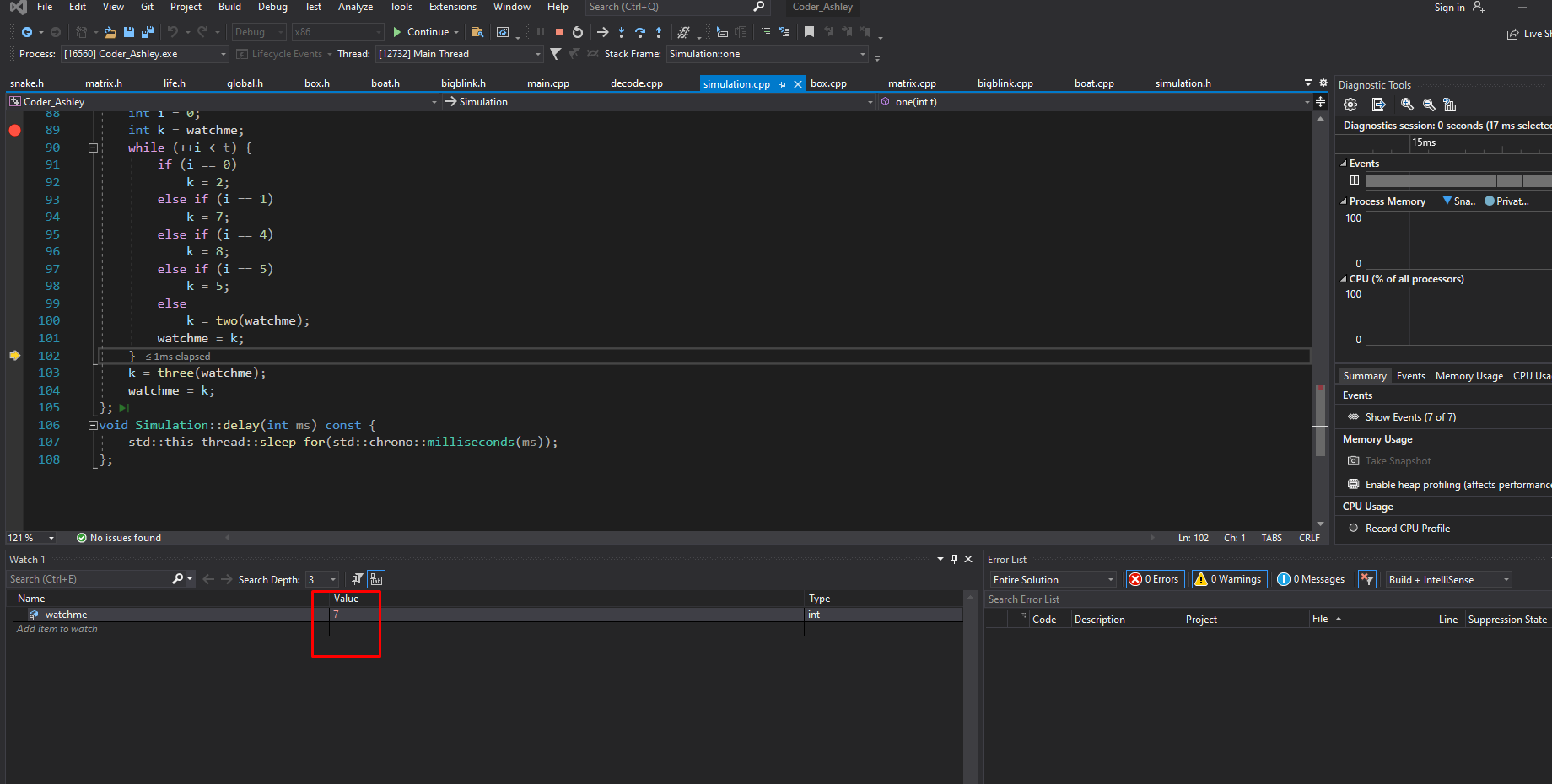
****

**Figure. Screenshot after placing a breakpoint**

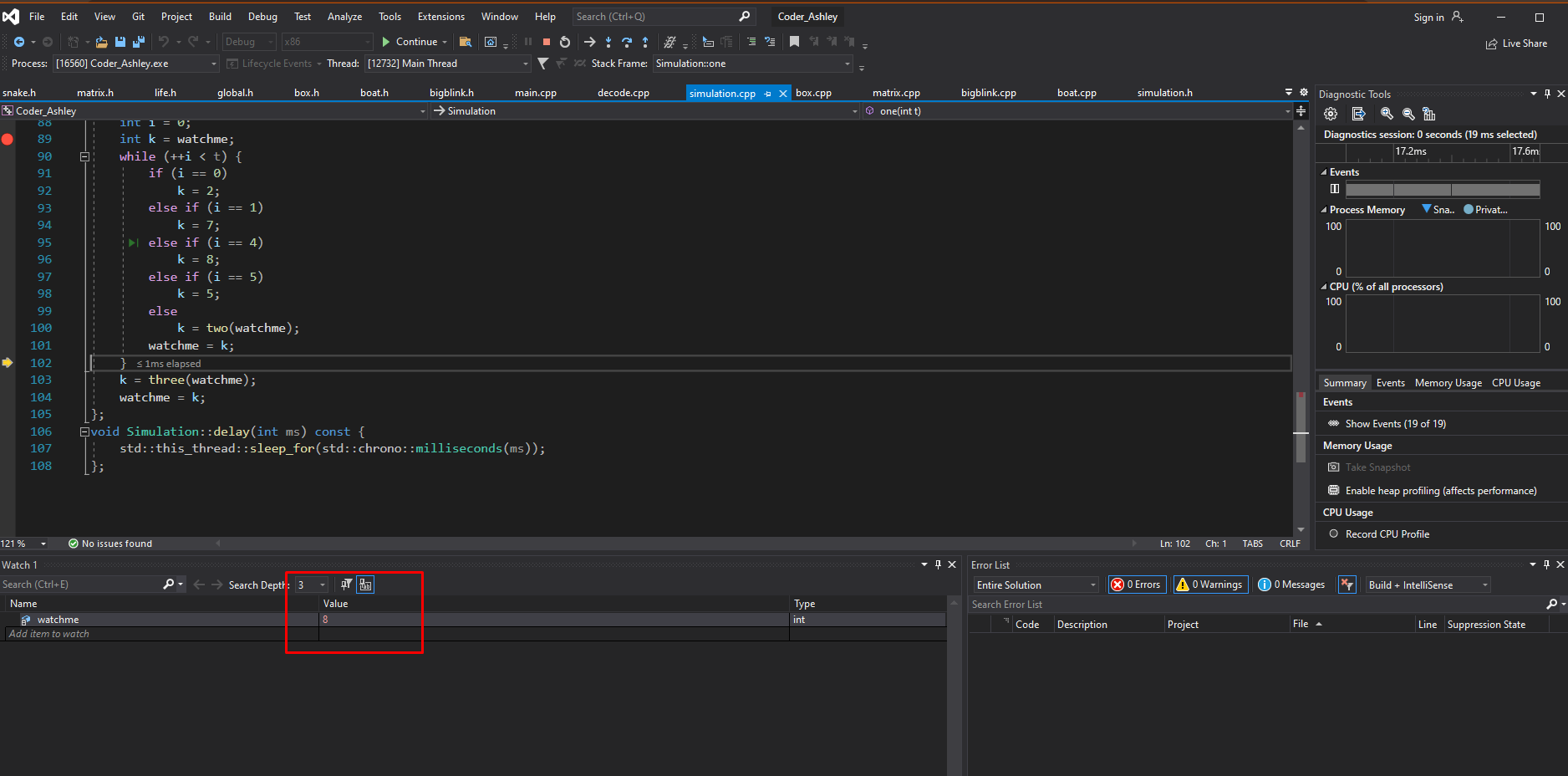
* Step through the rest of the code and make note each time Simulation::watchme changes in value. Do this until you find yourself in the main function.



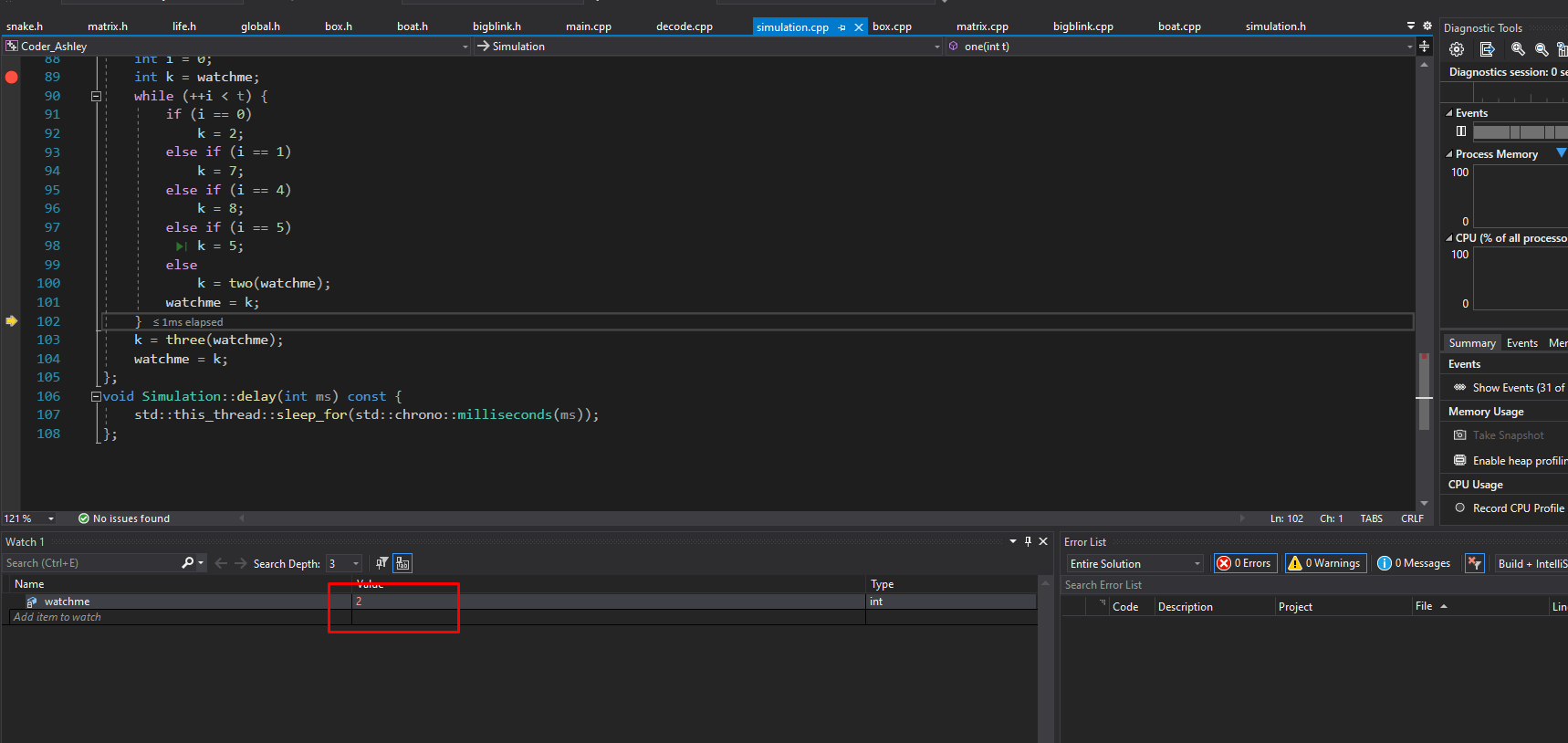
**Figure. Screenshot of watchme = 1**

****

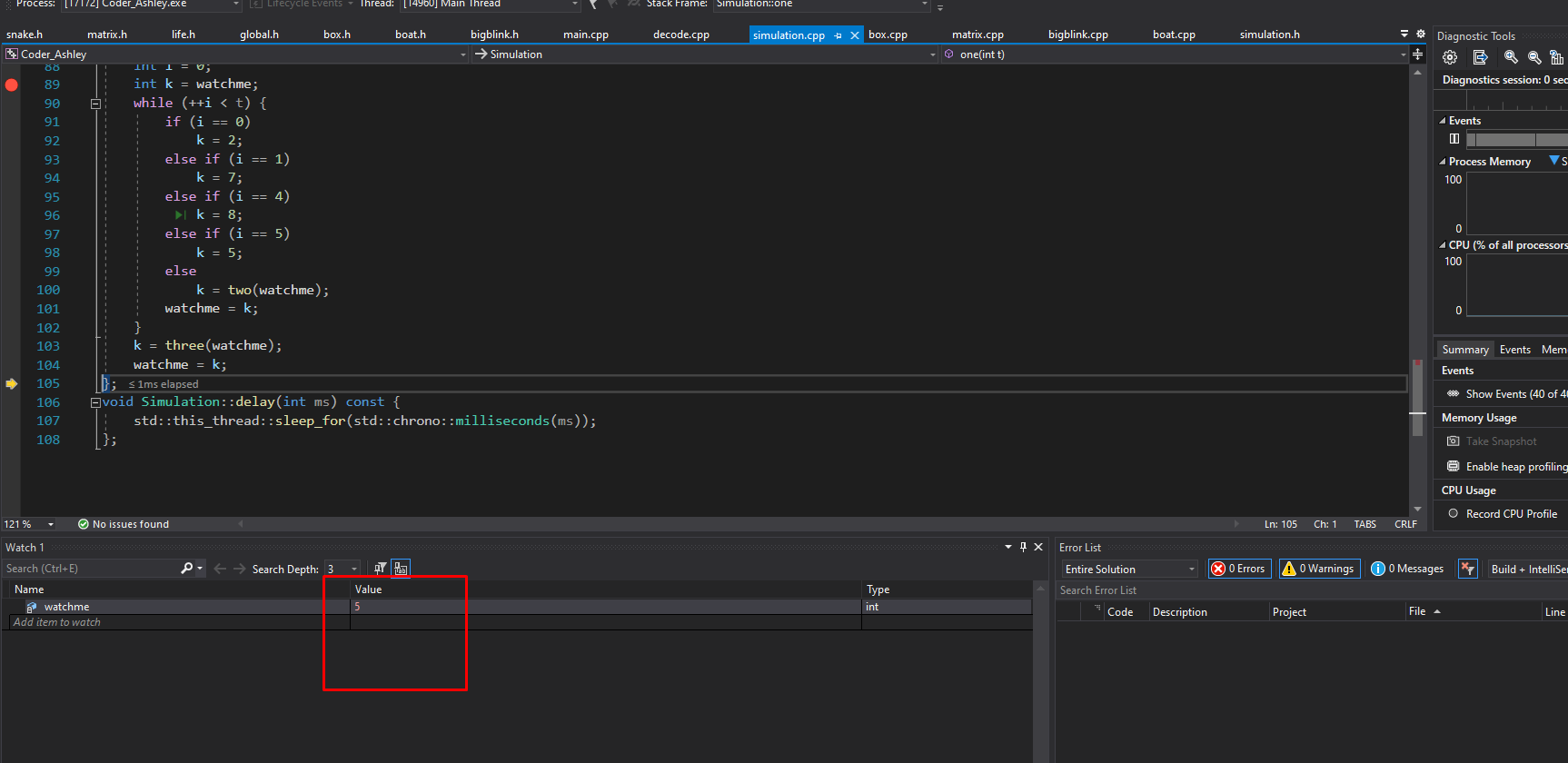
**Figure. Screenshot of watchme = 7**

****

**Figure. Screenshot of watchme = 8**

****

**Figure. Screenshot of watchme = 2**

****

**Figure. Screenshot of watchme = 5**

**Part 2: Analysis - 2:**

**Question:**

**After separating into individual files: If we replace your main.cpp file with the following, the program must build successfully. Note there are no mistakes here, this is a valid test case. If this main ends up not compiling, what might be the underlying issue?**

*#include "simulation.h"*

*#include "simulation.h"*

*#include "matrix.h"*

*#include "matrix.h"*

*#include "life.h"*

*#include "life.h"*

*#include "box.h"*

*#include "box.h"*

*#include "snake.h"*

*#include "snake.h"*

*#include "bigblink.h"*

*#include "bigblink.h"*

*#include "boat.h"*

*#include "boat.h"*

*#include "global.h"*

*#include "global.h"*

*int main(){}*

**Answer :**

* Incorrect Header File Names: Ensure that the names of the header files in the #include directives exactly match the actual filenames (including capitalization) and file extensions (usually .h for C++ header files). Any deviation from the correct filename or extension will result in a compilation error.
* Syntax Errors in Header Files: If any of the included header files contain syntax errors or other issues, it can lead to compilation errors. Check the content of each included header file for correctness.
* Circular Dependencies: If there are circular dependencies between the header files (e.g., Header A includes Header B, and Header B includes Header A), it can cause compilation errors. Circular dependencies should be resolved using forward declarations or by reorganizing the code.
* Incorrect Header Guards: If any of the included header files have incorrect or missing include guards, it can lead to redefinition errors when included multiple times. Make sure that each header file has proper include guards to prevent multiple inclusions.
* Build Configuration: The build configuration in your development environment may affect whether the compiler can find the header files. Ensure that the project settings or build configuration is correctly set up to include the necessary directories and files.
* Compiler-Specific Behavior: Different compilers may have slightly different behaviors when it comes to handling header files and includes. Ensure that the compiler being used is configured correctly.
* File Locations: Confirm that the header files are located in the specified directories or paths that the #include directives reference. If the files are not in the expected locations, the compiler will not find them.

**Part 2: Analysis - 3:**

**Question:**

**After separating into individual files: If we replace your main.cpp file with the following, the program must not build successfully. If this mainly results in a successful build, why might that be?**

*#include"simulation.h"*

*int main() {*

*Matrix m;*

*Simulation s(nullptr, 0);*

*}*

**Answer :**

* Dependency Chain: If matrix.h is included indirectly through other headers included in the project, the compiler might follow the dependency chain and include them automatically. For example, if simulation.h includes matrix.h and simulation.h is included in main.cpp, the necessary headers could be indirectly included.
* Header Guards: If the header files matrix.h and simulation.h contain proper include guards (e.g., #pragma once or traditional #ifndef and #define guards), the compiler may recognize that they have already been included elsewhere in the project and avoid duplicating the includes.