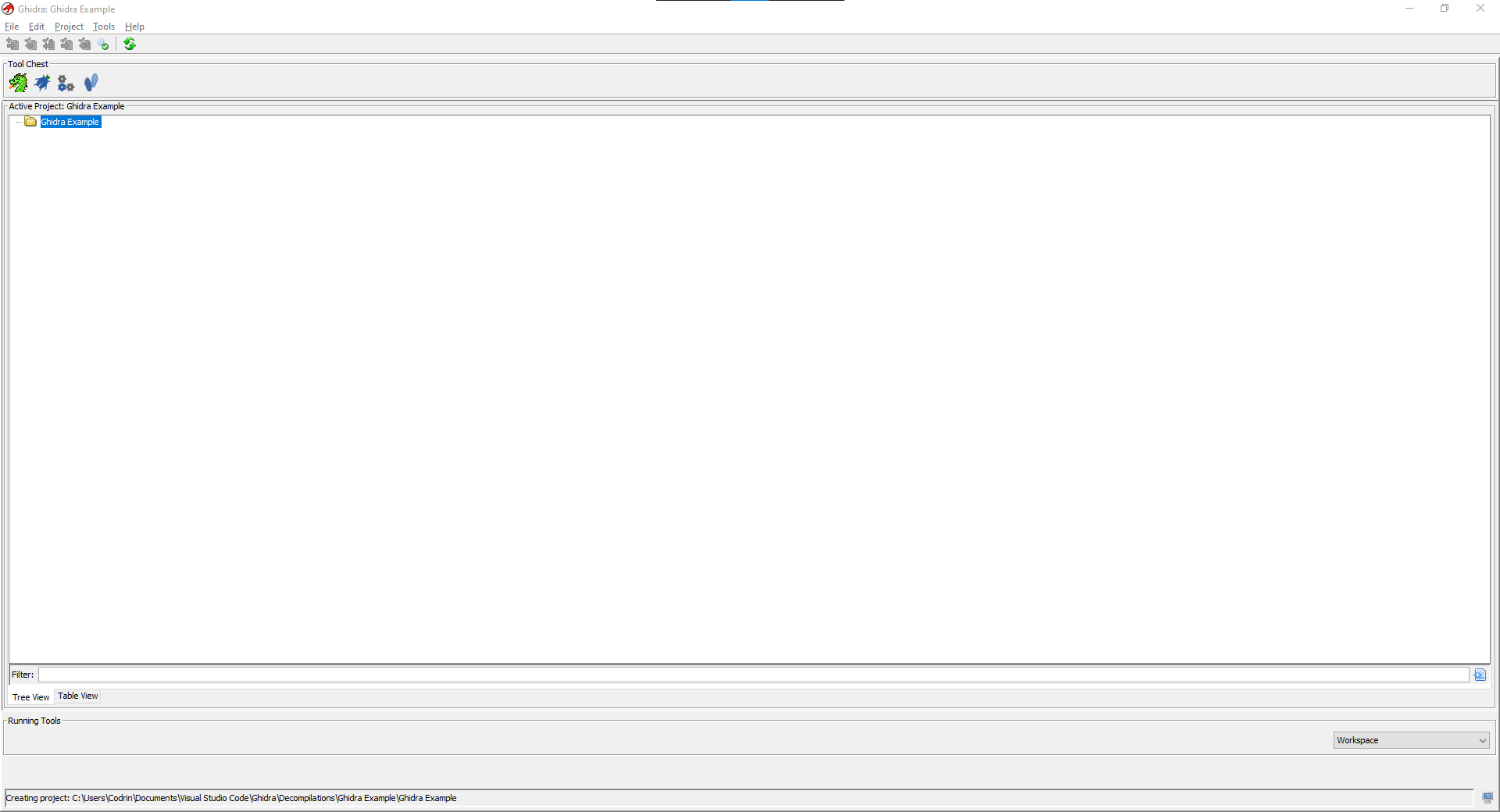
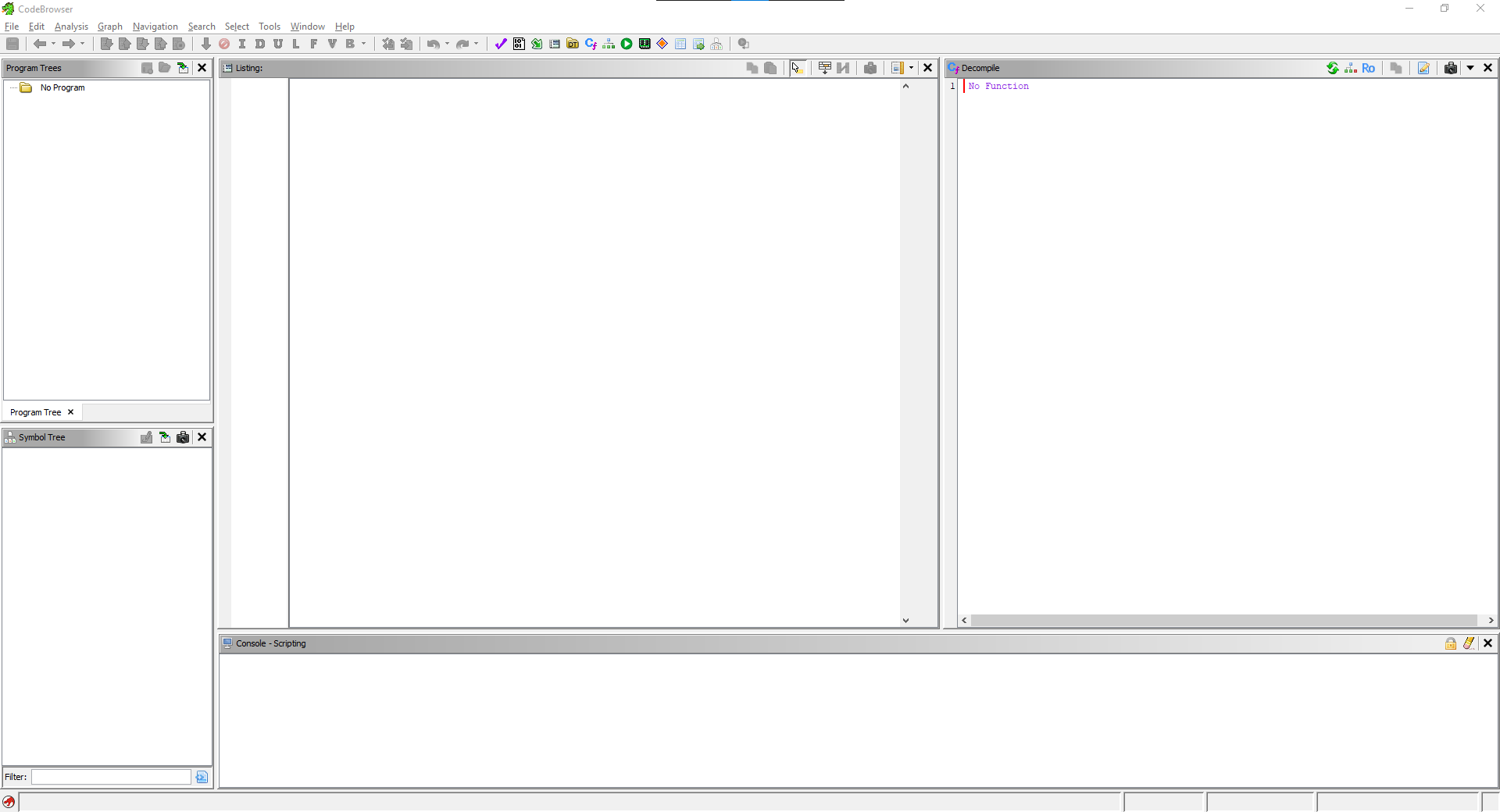
Ghidra, as an application, comes with its own documentation and help files. These help files include examples for using Ghidra and a detailed explanation on all steps to take.

This is an example from within the documentation, where the disassembly of an executable file "WinhelloCPP" is shown.

To begin, we will open up ghidra, and create a new project titled Ghidra Example

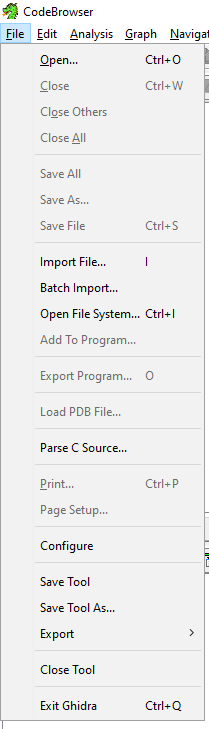


Inside of our new workspace, we can open the CodeBrowser tool



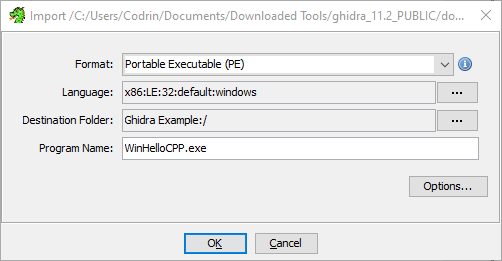
Then, we will have to import our file.

This is done by moving to File,



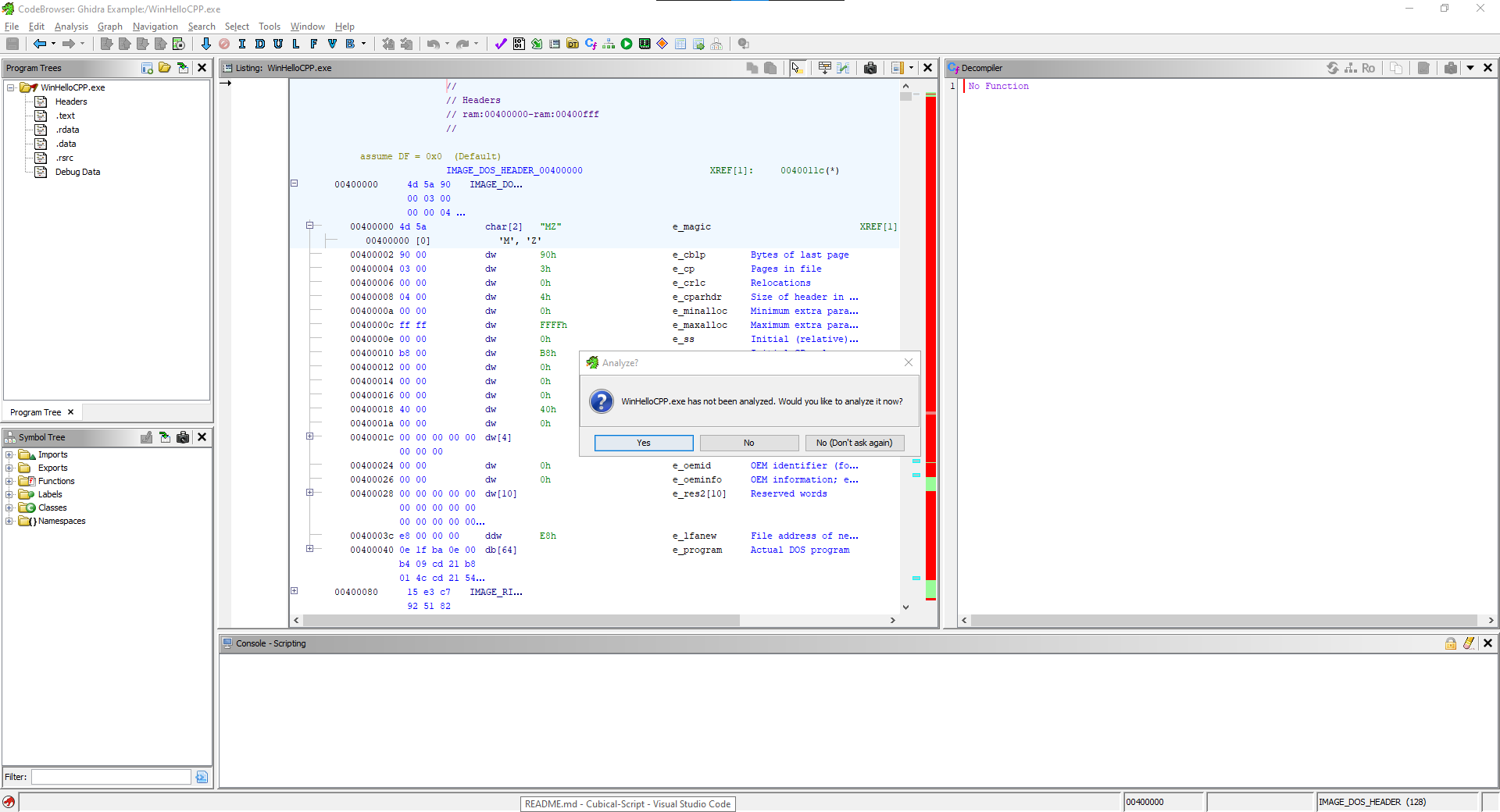
And selecting the file to import.

To initiate a disassembly, you must first specify the language of the file,

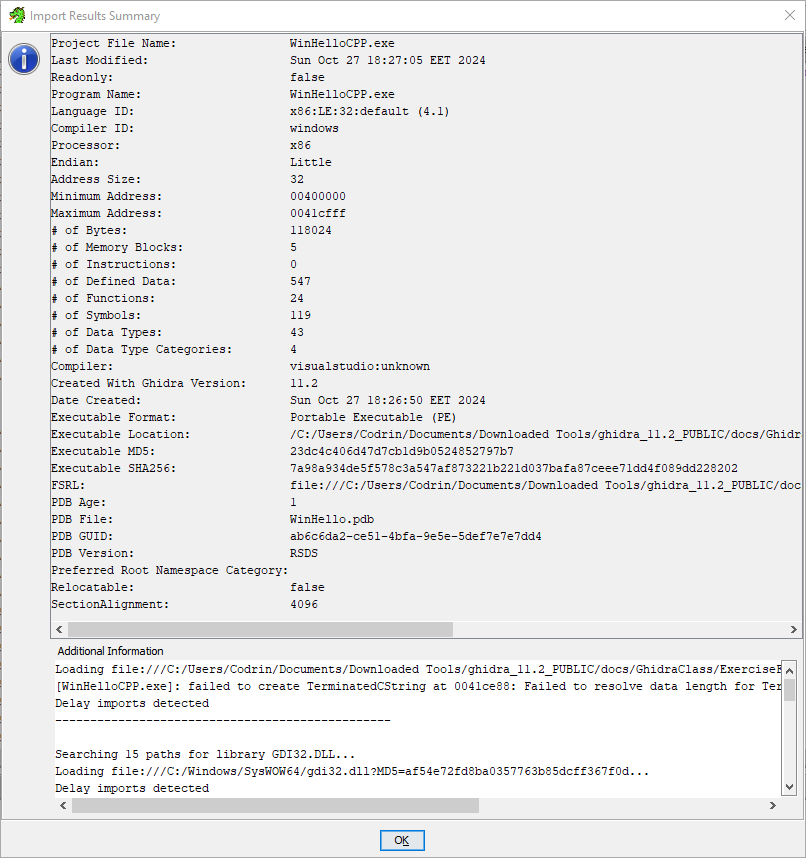


Ghidra usually detects it for you, but in the rare case it gets it wrong, the dropdown menu has all usable languages.

Click Ok to start disassembling the file. Once the process is finished, you will be prompted to preform an auto analysis,

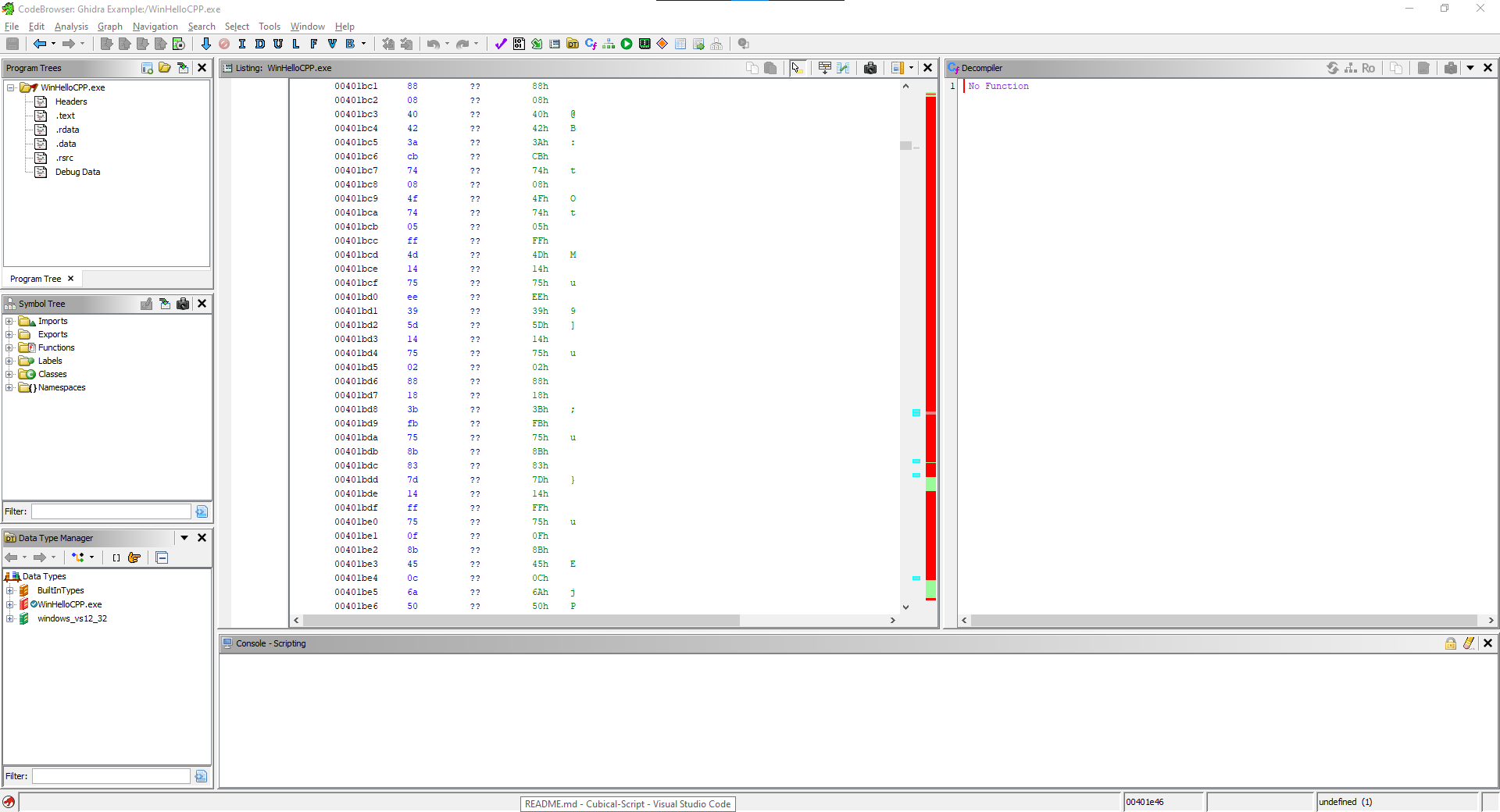


For the purposes of this example, we will not perform an auto analysis just yet. After closing the prompt, you will be shown the import results.



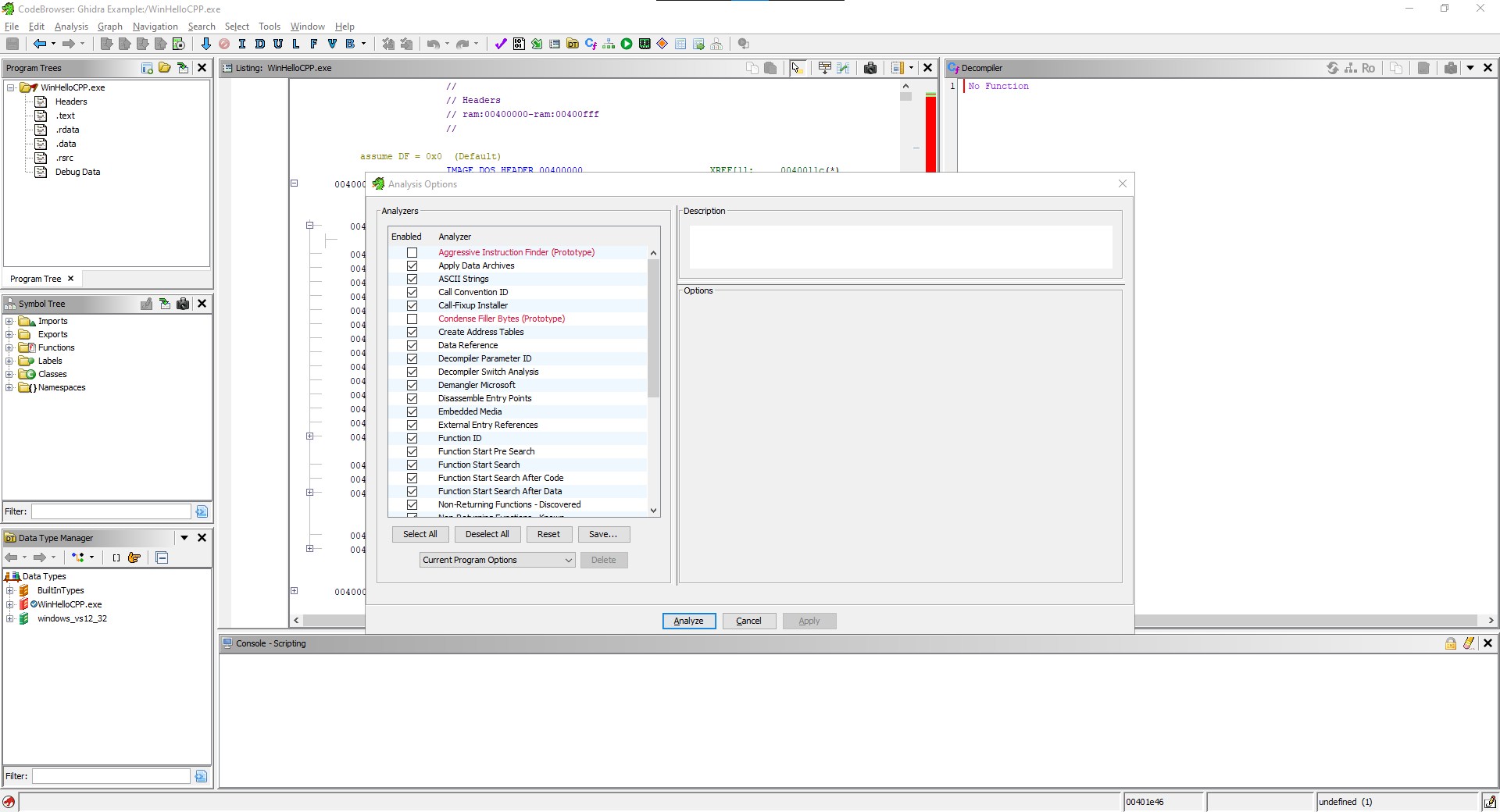
Closing these results, you will be able to see in the middle of the CodeBrowser the assembly code of the imported executable,

The bar to its right, is the navigation slider, coloured red are unrecognised fields, being fields that ghidra could not recognise the contents of during disassembly.



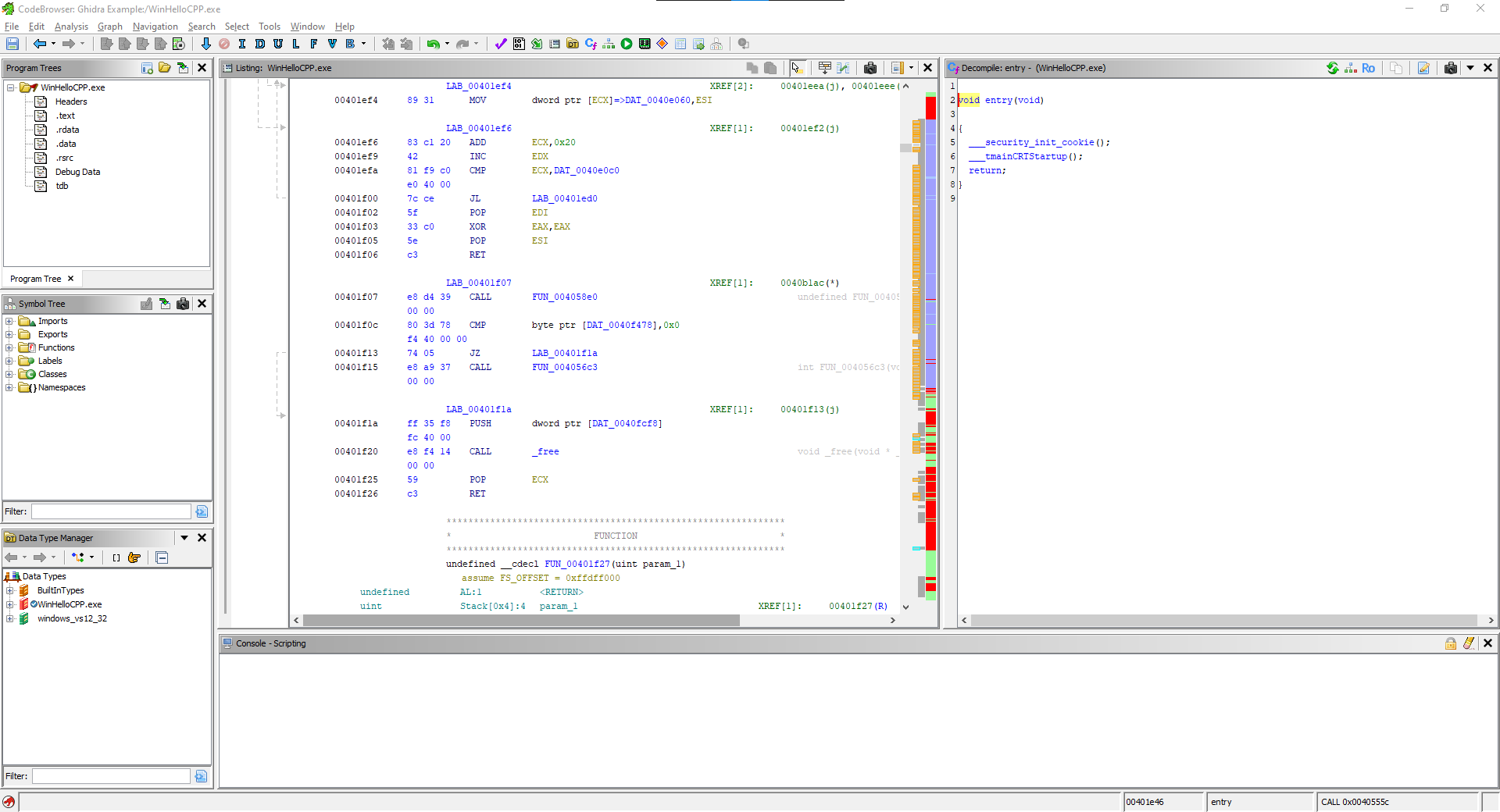
We will now request an auto analysis to display the changes it makes. This is done by clicking auto analysis inside of the Analysis tab, this request will bring up an analysis window, in which you can choose what you want to be done during analysis.

The configuration shown below is the default selection.



After the analysis is complete, you will be moved to the programs entry sequence, the starting point of the executable.

The code decompiler window at the right of our assembly code is a C representation of the actions performed by the entry function.

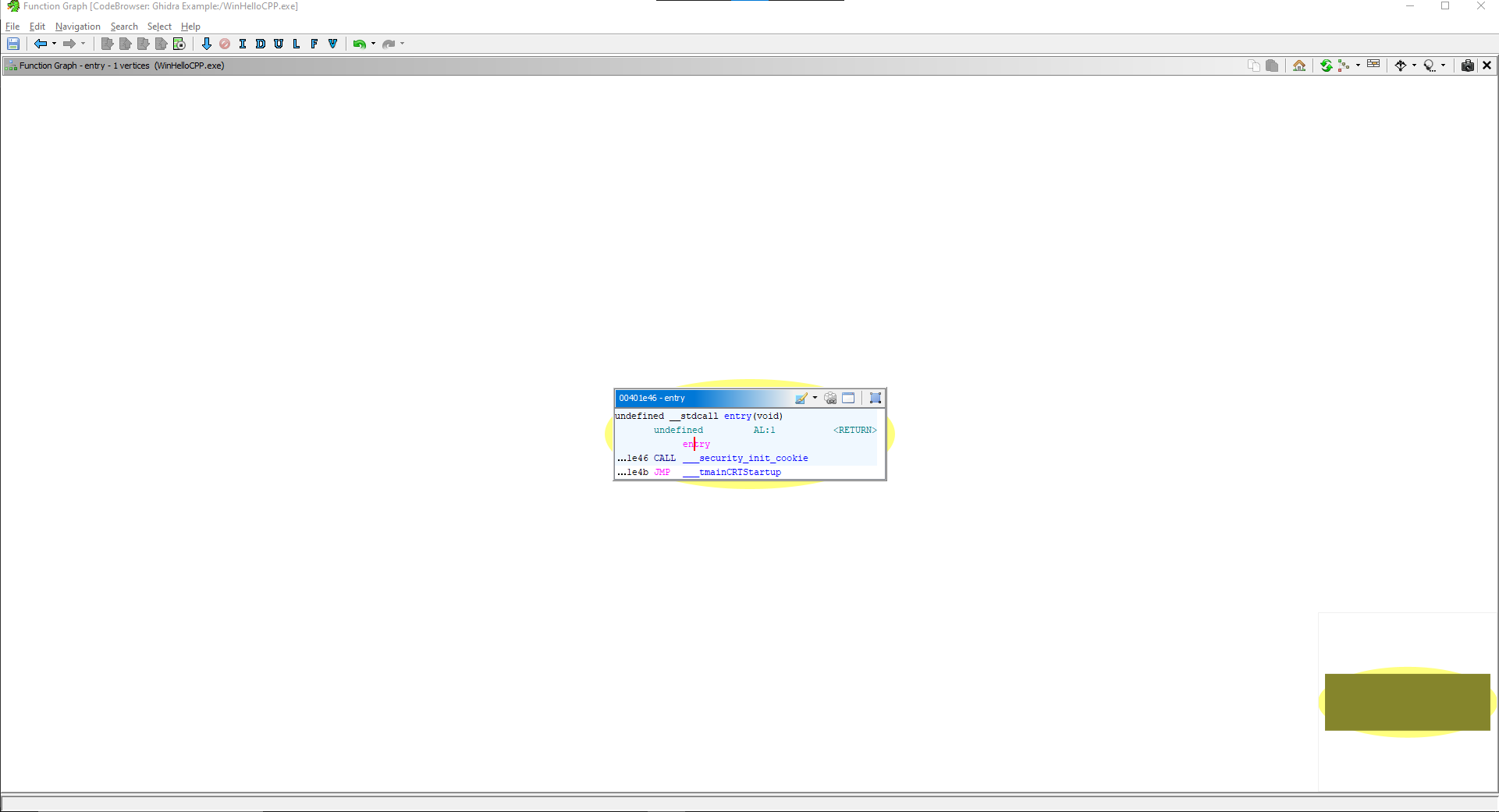


Additionally, you will see that the scrollbar has gotten populated with more colours, though the red sections are still visible, this is because sequences of code can remain unrecognised, even after analysis.

To not make this example too long, I will be covering the graphing tool inside of the code browser.

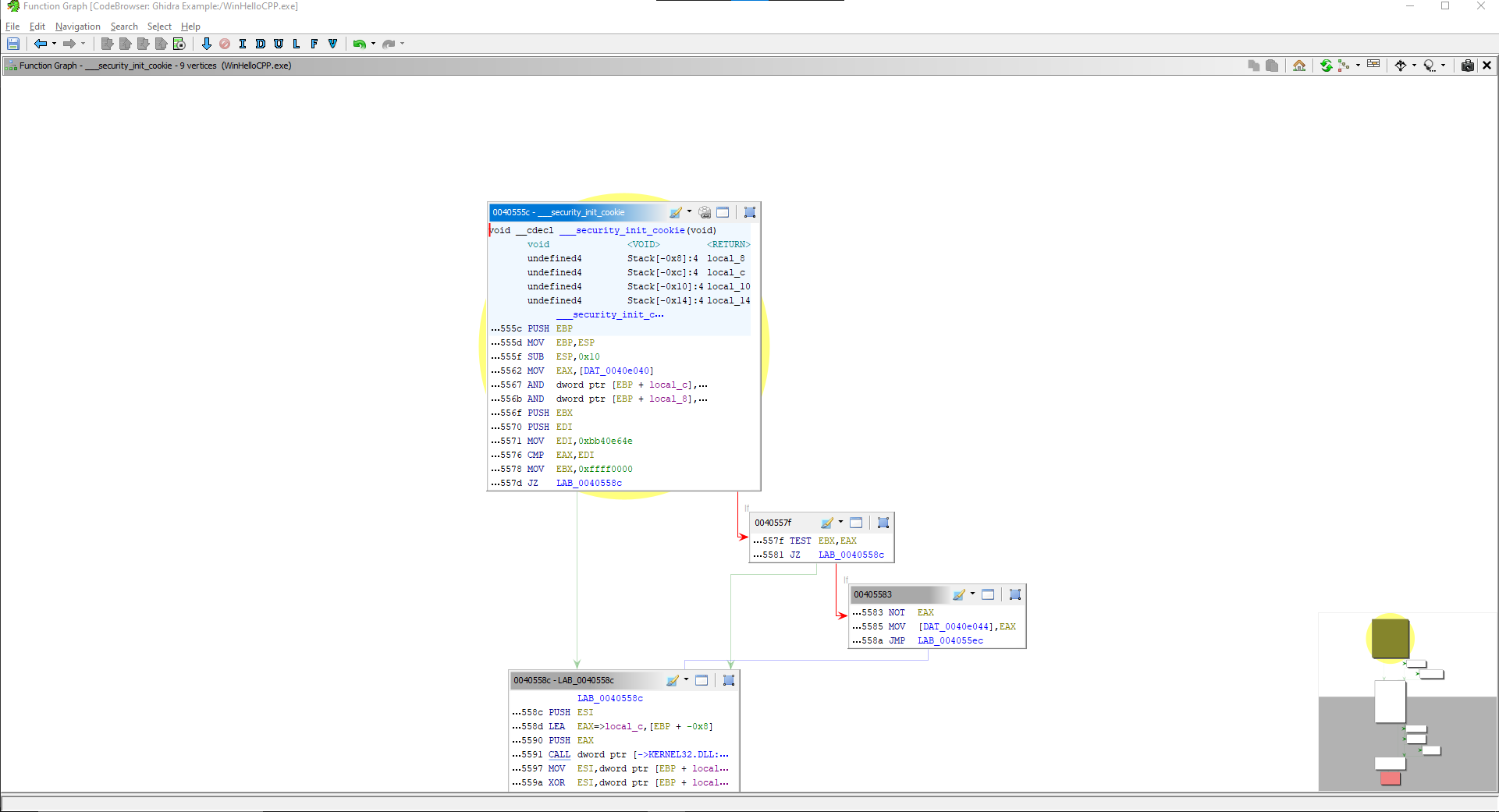
By clicking the graphing tool, you get a screen popup that shows the actions performed inside of the selected field of code.

Since I have selected the entry, this single box represents the actions performed when starting the executable.



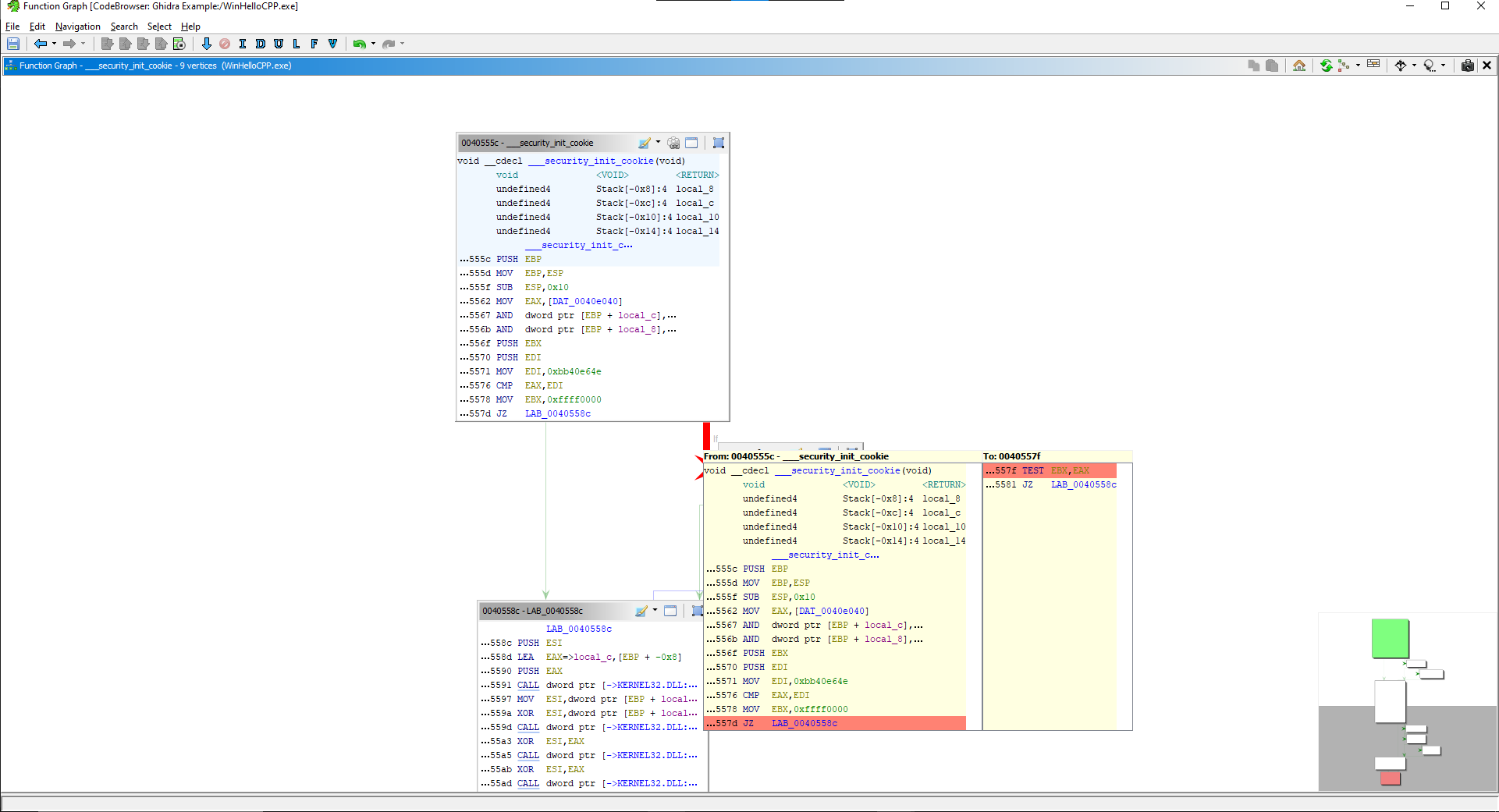
As you can see, this entry calls for a function.

Clicking on this, we're taken to the contents of this secondary function



The lines going out represent subfunctions, such as if, and the pointer returns to the main function once finished.

By hovering over a field, you will see the direction in which these sub functions are performed and by hovering over a line you will see the section being hovered on



Conclusion

This example is to show an overview of ghidra's code browser, tho this explanation was short, it showcases how powerful this tool is and how versatile it can prove to be