CS50 Bugs and Debugging

Overview

A **bug** is an error in code which results in a program either failing, or exhibiting a behavior that is different from what the programmer expects. Bugs can be frustrating to deal with, but every programmer encounters them. **Debugging** is the process of trying to identify and fix bugs that exist in code. Programmers will often do this by making use of a program called a **debugger**, which assists in the debugging process.

Key Terms

- bugs
- debugging
- debugger
- breakpoint
- `debug50`

Debugging Basics

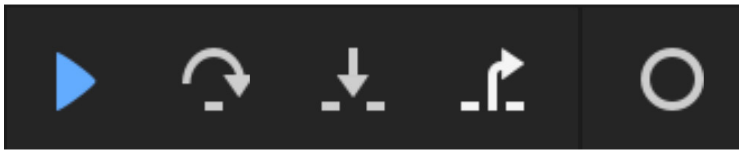
Programs generally perform computations much faster than a human possibly could, which makes it difficult to see what's wrong with a program just by running it all the way through. Debuggers are valuable because they allow a programmer to freeze a program at a particular line, known as a **breakpoint**, so that the programmer can see what's happening at that point in time. It also allows the programmer to execute the program one line at a time, so that the programmer can follow along with every decision that a program makes.

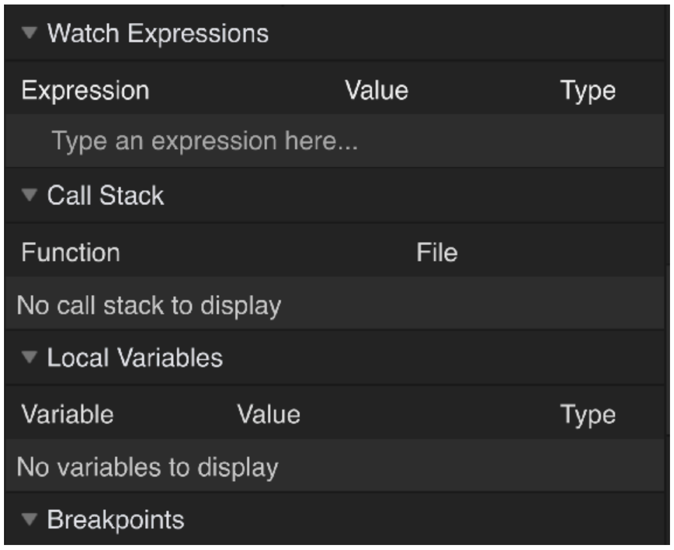
Using `debug50`

`debug50` is a program that we've created that runs a built-in graphical debugger in the CS50 IDE. Before you run `debug50`, you must set at least one breakpoint. If you have a general sense for where your program seems to be going wrong, it may be wise to set a breakpoint a few lines before there, so that you can see what's happening in your program as it moves into the section that you believe is causing the problem. If you're not sure at all, then it's totally fine to set a breakpoint at the first line of your `main` function so that you can step through the entire code from the beginning. To set a breakpoint, click the space to the left of the line number of your program. You will see a red dot appear in that space and it will be added to the list of all your breakpoints at the bottom of the graphical debugger window. You can remove a breakpoint by clicking on the red dot next to the line number.

Once you're satisfied with your breakpoints, run your program with as usual with `debug50 ./program_name` and any command-line arguments. Your program will run, and automatically stop at any breakpoints. In the debugger window, you'll see several tabs you can interact with.

At the top of the window, you'll see five buttons. The first, the blue triangle, will run your code until it hits the next breakpoint. The next button, the curved arrow, allows you to skip over a block of code. Next to that is the down arrow which allows to move through your code slowly one line at a time. The last arrow allows you to step out of a function (other than `main`). The last icon, the outline of a circle, clears all of the breakpoints that you set in your program.





The window at left summarizes some some features of `debug50`. In the "Watch Expressions" tab, you can type in a variable or function you want to watch while your code runs.

The "Call Stack" tab displays what function the line of code you are on is in and the file path for the programming that is running.

The "Local Variables" and "Breakpoints" are pretty intuitive. For each variable, you can see it's name, value and type. You can even override variable values by clicking on the value and typing in a new value. For each breakpoint you've marked, you'll see the file name, line number, and what appears on that line of code. You can turn off breakpoints from this tab by clicking on the checkbox next to the breakpoint, or delete it by clicking on the 'x' in the top right corner of the breakpoint when you hover over it.