CSE 274 – SPRING 2015 – INTERACTIVE DEBUGGER EXERCISE

[AUTHOR: DR. BO BRINKMAN]
This exercise is designed to help you learn the debugger. For that reason, I recommend that you do all the exercises in order. Skipping around, reading ahead, or finishing the exercises as quickly as possible is not going to be as helpful.

Write your answers in a Word document called "Debugger Lab" saved in the folder HW05Phase01 and commit the document to SVN. I will be grading your work only on whether or not you made a genuine effort to complete the exercise and answer all questions.

Step 1: Create a project file, and add sierpinski.cpp to the project. Also, look at correct-triangle.html to see an example of correct output. Read the source code for sierpinski.cpp (including comments), familiarize yourself with the options under the Debug tab of Visual Studio, and try to answer the following questions:

1.a) What is the purpose of this program?
1.b) Does the program produce output? If so, what kind of output is it, and where does it end up?
1.c) Does the program accept input? If so, what kind of input, and how is it provided?

Step 2: Build and run Sierpinski. An output file called "triangle.html" should be placed in your project directory, along with your source code. The file should exist, but you will notice that its contents do not look much like the correct output.

2.a) List all the ways that you can see that the current output of your program is different from the expected output.

Step 3: The first problem I noticed is that I expected the main part of the print-out to consists only of spaces and 8s, but the output I actually got has some strange characters in it.

The characters to print out in the main part of the file are stored in a big array called “grid.” The program is broken into three main parts: (1) Setting up the problem, (2) Filling out “grid,” and (3) Printing “grid.”

3.a) Explain one or two methods you could use to determine which of the three parts of the program is causing this first bug. (In step 4, I will tell you how I would do it.)

Step 4: I want to determine when the trash starts to appear in "grid." In order to do this I will set two breakpoints: One at line 85 (the last line of the "Setting up the problem" section), and one at line 115 (the first line of the "Printing grid" section). To be able to see line numbers, go to Tools > Options > Text Editor > C/C++ / General, and check the box for "Line numbers" under "Display".

Next, I will hit the green play button in my toolbar, to "Start Debugging." This can also be found under the Debug menu.

When your program stops at the first breakpoint, select the Locals tab in the window bellow the code, to the left, and scroll up until you find the variable "grid" … you shouldn't have to go up far. Right-click on "grid", and select Add Watch. This will open a window that shows you the variable name, its value (which is what the variable contains right now), and its type.

4.a) What would you expect grid to contain before the bug starts hopping? What does it actually contain at this point?

Hint: In order to get a better look at the contents of grid (which is an array of characters), in the Watch window, select the drop-down box at the end of the "value" column and choose "text visualizer." This should pop up a window that gives you view of everything contained in grid. The view in the Watch window cuts off when it runs out of room.

If everything was as you expected in question 4.a, then hit the play button again to continue debugging. Repeat the Watch process (inspecting the grid variable) when you hit the next breakpoint.

4.b) Based on your observations so far, which of the three parts of the program (setup, making the triangle, printing the triangle) must contain our first bug?

Step 5: In the previous step you should have identified that the bug is occurring in the *setup* phase of the program. The evidence is that, at the end of the startup phase, the array called "grid" contains lots of weird looking characters, instead of spaces, as we would expect.

5.a) Which lines in the code look like they should set the grid to be full of spaces?

Step 6: Once you have found the line that says "grid[i] = ' ';", it may or may not be obvious to you what the bug is. The bug is of a type that is VERY easy to over-look, and probably most of the class doesn't see it. Imagine that you can't find the bug just by looking at the code. Next, we will use interactive debugging to narrow things down a bit more.

First, stop the debugger (by hitting the stop button), and then remove all your old breakpoints. The fast/easy way to do this is to go to the Debug menu, and select Delete All Breakpoints. Next, we will set a breakpoint on line 65, the line that starts "grid[i] =". Then go ahead and run the debugger again.

At first it looks like this code should work. The for loop seems to be using its counter variable i correctly, and we seem to have the correct test condition in the loop. So, since we don't see the problem, lets try just stepping through the program to see if what we expect to happen actually happens.

The "step over" command in the debugging menu is designed to make the program execute the current line, and then stop on the next line of code that would be executed. In a loop, for example, each step should move you one line down the body of the loop, then when you step at the end of the loop it should take you back to the top. You should step through the body of the loop one time for each iteration of the loop.

6.a) sideLength*sideLength = 40000, so how many times would you expect to hit the line "grid[i] = ' '" if you are just using "step over" repeatedly?
6.b) Try hitting "step over" repeatedly, and see if it does what you expected.
6.c) What is the bug?
(If you have not found the bug after 5 minutes, you are allowed to skip to the next step.)

Step 7: In the previous step you should have identified the bug: The for loop has a semi-colon after it! This means that the "body of the loop" was not getting executed over and over (as we desired), but instead was only executed once, when i was 40000.

Fix the bug by deleting the stray semi-colon. Then, re-build and re-run the program, to generate a new triangle.html. Look at it, and see how it looks.

7.a) What problems were fixed in the preceding steps? What errors can we see now?

Step 8: Well, the weird characters are gone. Now there are only spaces and 8s, so that is good … but now we can see that what is getting printed out is not very triangle-like. In fact, did you notice that each row just contains repeats of the same character? Some rows have 8s, some have spaces, but it seems like there are not any rows that contain a mix.

Repeat step 4, and try to identify which of the three stages of the program is causing this incorrect output. If grid only contains longs stripes of 8s, then there are probably still problems in the first two stages. If grid contains a mix of 8s and spaces, then the problem is likely in the printing.

8.a) What should grid look like at the end of the first stage? What does it actually look like?
8.b) What should grid look like at the end of the second stage? What does it actually look like?
8.c) Which section is the current bug in?

Step 9: At the end of section 1, grid was mostly empty, as we expected. At the end of section 2, grid contained a mix of 8s and spaces, which was also as we expected. Hence, we suspect that the bug must be in the printing section (roughly lines 115-122). Remember that we observed that each row printed the same character over and over again.

Can you find the bug? If not, try stepping through the doubly-nested loop. You can put your mouse over ANY variable to see what its current value is, so you don't have to use Watch unless you want to look at something big (like a whole array, or a class structure). As you step through, you should be looking for anything that could lead to the same character being printed over and over, and only changing from line to line.

9.a) What is the bug in the printing loop?

Step 10: In the previous step you should have discovered that the “grid[i*sideLength +i]” is incorrect … you really wanted “grid[i*sideLength + j]”. Make the fix, and re-run your code to see if we are closer to having correct output.

10.a) What is wrong with the picture now?

Step 11: It looks to me like we have a problem with the top vertex of the triangle. Why is it in the top right, instead of in the top center? Use the comments in my code to find the part that is supposed to set the top corner of the triangle to be in the center.

11.a) Where is the bug, and how should you fix it?
11.b) If I had not commented my code, how would you have discovered which variable to fix? What if I had used crazy variable names like “fooa, foob, fooc, food, fooe, foof” instead of “p1x, p1y, p2x, p2y, p3x, p3y”?

Step 12: In the previous step, you should have discovered that the line “double p3x = sideLength -1;” is incorrect. Instead, it should be “double p3x = sideLength/2.0;”. Make the change, and re-run your program to see how it affects the output.

12.a) What is still wrong with the output?

Step 13: Things are looking pretty good now. We have something triangle-like, and the top point is in the right place. Something still seems weird though … it is as if the little hopping bug prefers to jump down and to the right … there ARE some 8s in the left half of the screen, but almost all of the triangles seem heavily skewed to the left, and down.

13.a) Which portion of the code do you think is causing the problem? If you aren’t sure, you can try repeating step 4, as well as looking over the code.

Step 14: It seems most likely that the problem with the code is in the middle section, where the triangle is actually generated. From debugging, I convinced myself that we were printing out what was in grid, so the problem must be that grid is being calculated incorrectly.

Personally, I could not see the problem right away. When all else fails, I generally step through the program, and see if anything unexpected happens. Try setting a breakpoint on the first line inside the loop that builds the triangle (the line that starts "grid[((int)(cury+0.5)"), and start stepping.

Any time you have conditional logic (like a switch statement of if statement) within a loop, you might have to step through the loop many times before you see the problem … because the problem might occur in only one of the cases.

You have a couple of options now. One is to just step through the loop for a while, until you have seen all of the cases in the conditional logic work.

The other option is to create a break point within each condition. So, for example, put a break on "case 0", a break on "case 1" and a break on "case 2". Then run the program in the debugger. The first time you hit a particular breakpoint (say the case 2 breakpoint), step through the code and make sure it does what you expected. If so, you can remove that breakpoint, and continue debugging. This way you can try all three cases without having to do a whole lot of manual stepping.

14.a) What was the bug?

Step 15: In the previous step you should have discovered the problem was with the “case 0” case within the switch statement. Whenever you hit case 0, the bug hops halfway to point 1 … but then the program drops through into the “case 1” part, and the bug hops halfway to point 2, as well! This is why the bug seems to never hop left: Any time it hops left, it immediately hops to the right again. The fix for this is to insert the missing break statement, just before “case 1:”.

Make the fix, and re-run your program. If all went well, your program should produce output that looks very similar to (but not identical, because of the randomness involved) the provided correct-triangle.html.

Congrats!

Summary:
In this exercise you have learned to

1) Set breakpoints
2) Run the debugger
3) Step through your code
4) Inspect variables through tooltips and through the Watch window
5) Follow a pattern of thinking in debugging. Always try to:
    a. Predict what **should** happen
    b. Design a test to see if it **did** happen as expected
    c. A plan for what to do, based on the outcome of the test