Evan Tellep

Dr. Chundur

IT2090C

30 August 2016

Debugging: Lessons Learned

As programmers debugging is something we all have done, and it’s something we will always be doing. As a beginner program I can say pretty confidently that I have a good amount to learn about proper debugging techniques. Though I have been introduced to the debugger tools in the IDE’s I have used I really have not been introduced to better debugging practices and standards. I believe I have a lot I could learn from the article.

From reading the article right away I can tell you my debugging practices are lacking. Most of what I do is put in print, or other statements, thrown in the code semi-randomly to try to visualize where the error is at. This works sometimes, but as stated by the article is definitely not best practice. Print, and other statements, can greatly increase the time it takes your code to run. So far I do not find this to be a big issue, but there are certain instances where this could be a problem. There is also a great chance that, since you don’t know where the error is, it’s very possible that you will take more time putting in statements that are unneeded. It’s a highly inaccurate approach, and can be very time consuming. I feel if I implement the Scientific Method: Debugging Edition, I can more efficiently locate errors in my code rather than guessing.

Another thing that stands out to me greatly is my use of brute force debugging. Though it seems considered a valid form of debugging, I know that it can take a good amount of time. It can take much more time to take the approach of re-writing a program than just debugging it using good practices. There was a piece of code I wrote over summer dealing with calculating totients. I ended up re-writing that code four times because I would get so jumbled as I was debugging, because I wasn’t keeping track of what I was doing, that I just got frustrated and started over. In the end this approach worked, but I feel that if I were to have implemented some better debugging practices I wouldn’t have taken as much time to complete that project.

Sometimes, but rarely, I do run in the issue to where I don’t put enough psychological difference in my variable names. I do try to make my variable names unique and descriptive, but sometimes I fall into the trap of laziness and make similar variable names. I’m sure I would have a much easier time programming if I make sure to put enough psychological distance between the names of my variables.

Overall I found this article to be very informative. Debugging was something I’ve done before, but it was always something I knew I didn’t do to the best of my ability. Reading this article showed me a good amount of ways I can improve my debugging skills. I plan to implement the steps of debugging that are similar to the scientific method, practice debugging and logging my steps to avoid having to brute force my way through the problem, and put good psychological distance between my variable names so I don’t make simple typing errors that could cause me a lot of headache. This article was very helpful and I look forward to using these steps to help improve my debugging, and to lessen the frustration of the task.