Step 7: Debugging

/run/media/lijunjie/资料/Learning Video/writing-running-fixing-code/03_testing-and-debugging/02_debugging/01_step-7-debugging_instructions.html

Once you have found a problem in your code, you need to fix it—this process is called *debugging* . Many novice programmers (and even some moderately experienced programmers) debug in an ad hoc fashion—trying to change something in their code, and hoping that it will fix their problem. Such an approach is seldom effective, and often leads to much frustration.

Returning to our doctor analogy from earlier, suppose you were sick and went to the doctor. Does your doctor say “Oh I don’t know what is wrong, but try this medicine. If it doesn’t fix things, come back tomorrow and we’ll try another medicine…” If your doctor does treat you this way, it might be time to find a new doctor! Trying random “fixes” in the hopes that you will get lucky on one is not a good way to diagnose and fix problems. Even worse, if your symptoms change during this process, you have no idea if one of the random “fixes” you tried made things worse, or if your untreated condition is progressing. Similar analogies can be made to any profession that diagnoses and fixes problems (such as mechanics).

Hopefully, your doctor (and mechanic) follow a more scientific approach to diagnosing and fixing problems. As do they, so should you in diagnosing and fixing your program. In fact, debugging should be an application of the *scientific method* , which you may have learned in a science class. See the figure below: