Essentials → Software quality → Bugs and debugging

Theory: Bugs and debugging

4 minutes 0 / 2 problems solved

Skip this topic Start practicing

3284 users solved this topic. Latest completion was **about 11 hours ago**.

Current topic:

Bugs and debugging

Topic is required for:

Unit testing

Debugging in shell

Table of contents:

§1. What is a bug?

A **software bug** is a problem that causes a program to crash or produce invalid output. There are many reasons for software bugs; most common of them are human mistakes in software design, coding or understanding of the requirements.

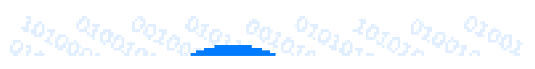
A program is usually called **stable** if it doesn't have a lot of obvious bugs. If the program has a large number of bugs that affect the functionality and cause incorrect results, it is considered **buggy** or **unstable**. A user cannot successfully interact with such software. It is important to find bugs **before** users start interacting with your program.

§2. Bug etymology

You've probably already heard the term "bug" and can imagine what it means. Here is a little story about this.

The first computer bug was discovered by Grace Murray at Harvard University in 1947, while she was working on Mark II computer. During the investigation of an issue, an actual moth was found between the contact of the relays. The moth was removed and added to the logbook, which now is located at the Smithsonian Institution's National Museum of American History in Washington, D.C.

So, computer errors are often called bugs. This term is used to describe any incorrect program behavior until the specific nature of the error is determined.



Some Internet sources give a different story of this term. You can google it if you are interested. Let us know if you find a story most reliable to you.

§3. Why do programs have bugs?

Developers often say that a program without bugs doesn't exist. There are some common reasons for bugs in software:

- communication issues in the team;
- misunderstanding of the requirements;
- software complexity;
- programming errors (programmers, like anyone else, can make mistakes);
- time pressure;
- use of unfamiliar technologies;
- an error in a third-party library (yes, that happens too).

During this course, you will mainly encounter bugs caused by misunderstanding of the given requirements or programming errors.

§4. Avoiding bugs

It is almost impossible to avoid all bugs in a large program, but it is possible to reduce their number. Here we give you five steps that can help to do that in both educational courses and industrial programming.

1. Make sure you know what to do. As a programmer, you need to understand the requirements of a program that you are going to work on. If you have doubts, you can always find some help on the Internet or among fellow developers.
2. Decompose a program into small units that are easy to look through and understand. A good architecture reduces software complexity, and, as a consequence, the number of errors.
3. Write easy-to-read-code and follow all the standards of the language. It will also allow you to make fewer errors.
4. Run the program with boundary input values. Do not forget to consider different cases: 0 or a huge number as an input value, 0 or 1 element as an input sequence. Such cases often reveal bugs.
5. Write automated tests that will check the program at the build time.

We will not discuss automated tests in this topic, but we will return to that later. At this moment, you can simply create a set of input values and run the program manually (as it was described in step 4).

§5. Debugging

Suppose you know that your program does not work correctly for some input values. To fix this bug, you need to find it in the code and then make some changes.

To locate a buggy place, you can:

- read the code and try to understand what it does for the input values;
- start the debugger and see the current values of variables and the control flow of the program;
- print the current state of the program in critical parts of the code (logging) and then analyze it.

The combination of the approaches above will allow you to find most of the bugs in your program.

Report a typo

263 users liked this theory. **0** didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (3) Hints (0) Useful links (1) Show discussion