How to Debug Programs

C/C++ Programming I & II

Debugging a program is a step-by-step logical process that first requires a full understanding of the following four concepts:

1. The program requirements.
2. How the code is intended to meet those requirements.
3. The specific purpose and behavior of absolutely every piece of code.
4. The expected value of every variable, other expression, and output at every point in the code.

Attempting to debug a program without this understanding is a pointless exercise in frustration and futility.

Regardless of the tools used for debugging, the general procedure is the same and consists of systematically stepping through the code to determine:

1. if statements are being executed in the expected order, and
2. if the values of relevant variables, other expressions, and outputs are as expected.

When errant code flows, errant variable or other expression values, or errant outputs are found, the cause(s) must then be determined and fixed in the order they occur. For programs that process input from the user or other source, the input being used when an error occurs should also be used during debugging.

Debugging Procedure for Beginners

Step 1:
Rather than starting off by debugging a program using the computer, it is often helpful for beginners to start with a printed copy of the code and proceed through it step at a time in the same order the computer is expected to run it. Each time a variable or other expression is encountered write its expected value next to it, and similarly for any print or other output statements that are encountered. This procedure often finds the problem or at least points out where you do not really understand something about how a piece of code actually works.

Step 2:
Once you have completed step 1 and have made any necessary code changes, if the program still fails it is time to use a debugging tool to step through the code as it runs and compare the values it reports with those you previously wrote on your paper copy. The easiest-to-use debugging tools are usually built into the IDEs themselves and instructions for using three popular ones are provided in the appropriate version of the "Using the Compiler's IDE…" documents on the course's Canvas website. If you do not have a decent debugger or are averse to learning to use one, you are doing yourself a major disservice.

In addition to a good debugger's ability to step through source code one statement at a time to follow program flow and check expression values, it also allows "breakpoints" to be set, which cause the program to pause at selected points for further examination. These two basic and easy-to-use features are sufficient for most situations and can be summarized as follows:

1. Step to the first statement in the code and check the values of any relevant variables, other expressions, and outputs. Are they as expected? If not, why not?
2. Continue to step through the code, making sure that the intended statements are being executed and relevant values are as they should be.
3. If there is a specific point in the code you are interested in inspecting and do not want to spend time single stepping to it, set a breakpoint on that line and let the program run to it.

© 1992-2021 Ray Mitchell