The Essential Elements of Debugging Software Programs

Wm. M. Stewart

14 November 1989

> Insects are small but their world is large, and they outbreed us all the time. *N. J. Berrill, Sex and the Nature of Things, 1953*

A computer program that produces incorrect results has a logic error—a bug. Unfortunately, automated prevention of all bugs is and will forever be impossible, for the machine can only execute your instructions, not your intentions. In other words, an automatic system might be able to identify syntax, format, and other 'obvious' errors, but a program that runs just fine and prints garbage has a logic error that only you can correct.

The debugging challenge is therefore first to minimize the number of bugs up front, and then to track down the ones that slip through as quickly as possible. The following guide is a checklist to assist you in both these activities.

1. Prevention

The first principle of bug prevention: understand the problem completely; write the solution top-down; desk-check the program with simple data; only then run the program on the machine for the first time.

Use meaningful variable names, consistent indenting, and generous horizontal and vertical space, for bug prevention is heavily dependent on program readability. Where more than one choice is possible, adopt a standard format for variable names, comment blocks, continued statements, subprogram headers, etc.

Use the shortest, simplest solution possible, for shorter, simpler programs have less bugs. (At the same time, break complex expressions in two, avoid tricky code, and in general favor clarity over brevity.)

Never write the same code twice: encapsulate any replicated process in a subprogram. Divide subprograms longer than a page into smaller logical units. Debug subprograms separately whenever test cases can be easily generated.

Initialize all variables, including arrays and pointers. Never use a variable for a second purpose, even if no longer required for the first purpose (exception: elementary loop counters).

Verify in the code that input data has the correct format. Verify in the code that subprogram parameters have the correct format. Echo all input data.

Parenthesize expressions when in doubt about operator priority. Explicitly convert variables of different data-types to the same data-type when combined in the same expression.

Include code in decision structures to print an error message for the excluded case.

Beware of floating point round-off. Use the highest floating point precision possible.

Use a debugging compiler when available. Activate all debugging aids like subscript checking and overflow detection. Be aware of default declarations.

2. Cure

Write a dump routine to print the program variables with labels: either write a subprogram which accesses the variables globally, or write the dump routine as a block of code in the main program and then copy it from place to place with the text editor. If using a subprogram, pass a text parameter to title the dump—usually the location of the call.

Track down each bug with the binary search algorithm: dump the variables at the beginning of the program; copy the dump forward until the first dump prints correct data and the second dump prints incorrect data; repeatedly divide this interval in half until the location of the bug has been narrowed down to one line.

Dump the variables before and after decision structures, before and at the end of loops, before and after subprogram calls, and at the beginning and end of subprograms, as required.

Reduce debugging output by first tracing the flow of control through nested, high iteration structures before invoking the dump routine, with statements like:

```
print 'debug - while loop 1'
print 'debug - for loop 2 - J =', J
print 'debug - if block 3a'
```

Data-dependent debugging can be use to reduce debugging output or to trap anticipated bugs:

```
if DebugSwitch then
   call Dump ( 'debug - start search' )
if K > 100 then
   call Dump ( 'debug - for loop 4' )
if Count < 0 then
   call Dump ( 'debug - negative Count' )
```

Always debug from a current listing. Timestamp all output.

Fix the earliest error first. Only fix one bug at a time. Check the simplest possibilities first.

Mark debugging statements with the string '`debug`', enabling easy location with the text editor.

Comment out debugging statements not currently required instead of deleting them.

Finally, stay calm. Program debugging is excellent exercise for the rational mind, and the capture of each bug is its own reward. Good luck, and good bug-hunting!