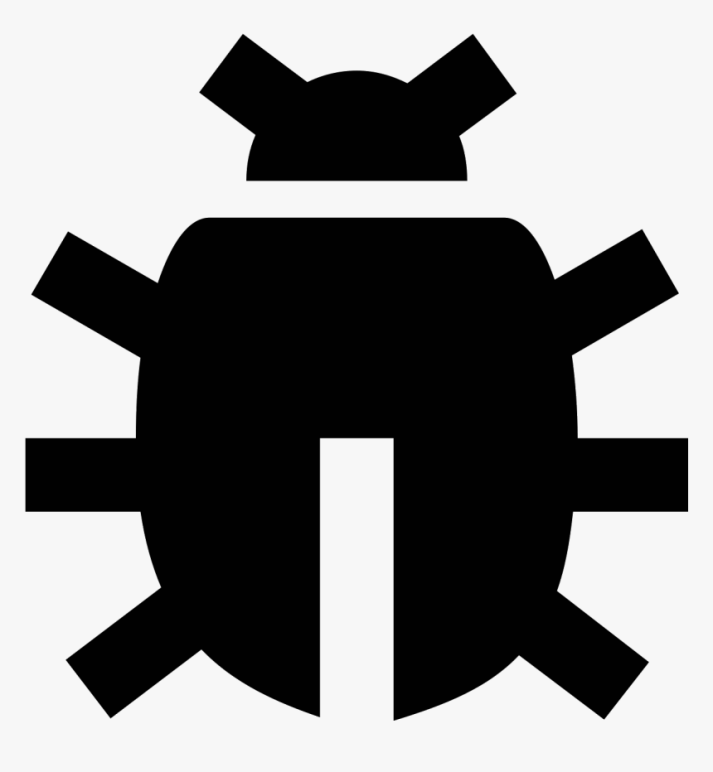
Debugging

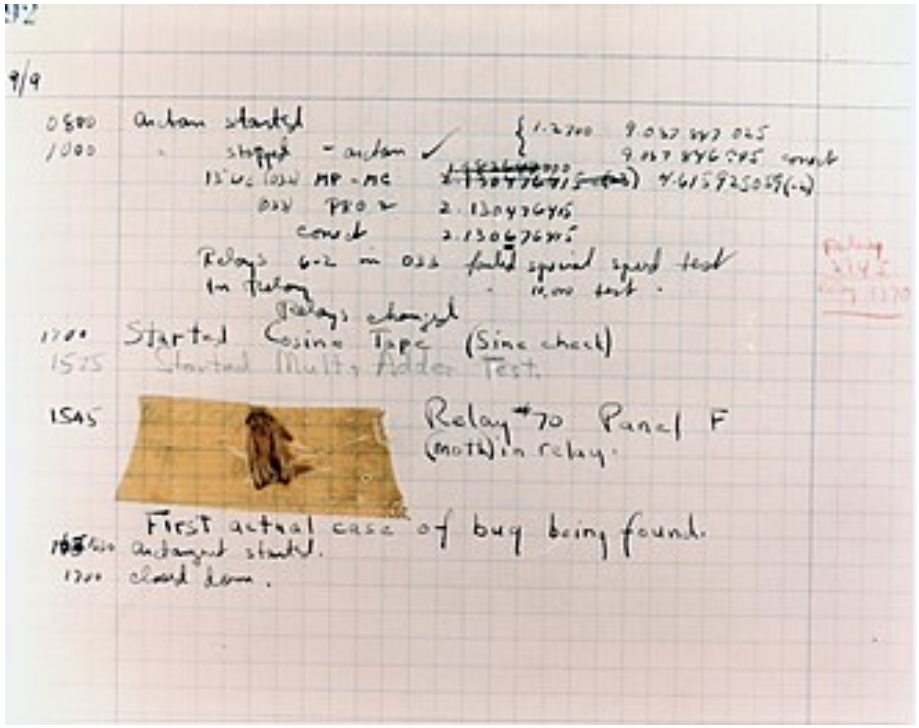
Yusuf Çelik

June 15, 2022

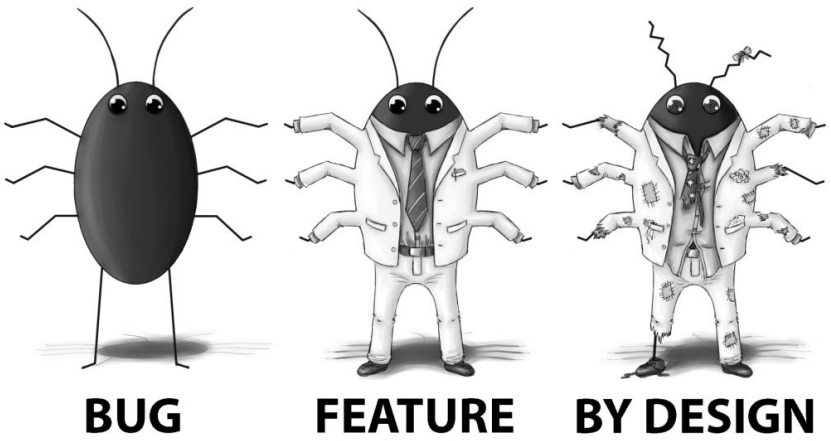
What is a bug?



Harvard Mark II

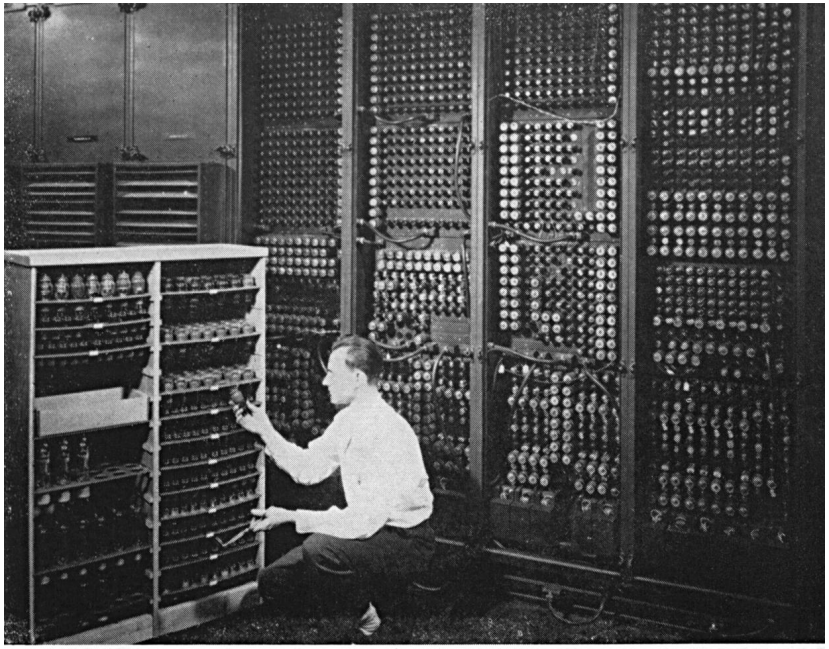


Is it really a feature?



But how can we debug our programs?

What debugging methods do we have?



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

High Level solution and Testing

- design your project in a clean fashion
- Test your program as much as possible

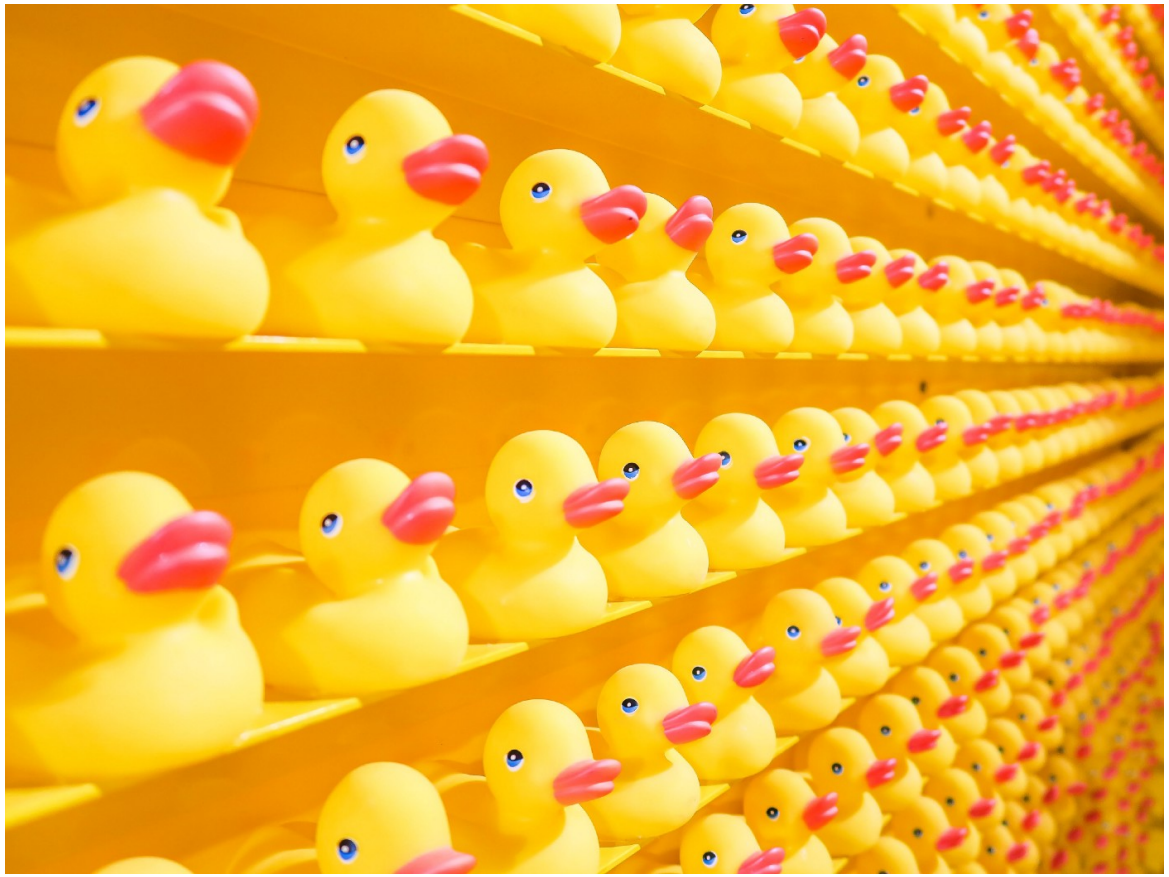
Read compiler warnings

Logging

AKA print debugging

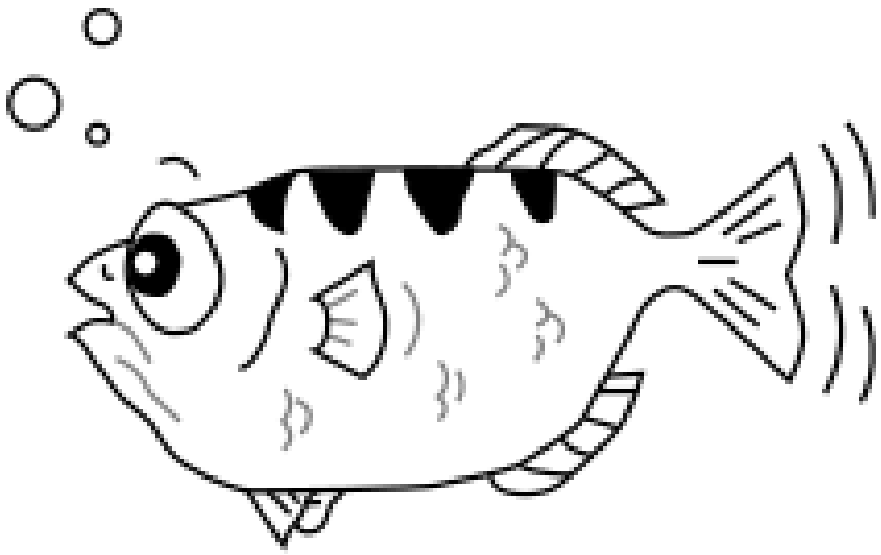
```
# Object name
o medallion.obj

# Begin list of vertices
v 1 -22.9474 7.70877 1.0
v 4.48816 -12.596 3.1349 1.0
v 1 -13.992 0.524152 1.0
v -1.30385e-08 -13.992 0.524152 1.0
v -5.21541e-08 -22.9474 7.70877 1.0
v 1 -16.3405 6.8999 1.0
v 3.00765 -11.2079 -0.243105 1.0
v 9.31323e-09 -16.3405 6.8999 1.0
v 1 -20.9952 -1.95239 1.0
v -2.04891e-08 -20.9952 -1.95239 1.0
v -4.84288e-08 -9.15876 8.99448 1.0
v -4.65661e-08 -15.1243 10.3649 1.0
v 2.23517e-08 -11.6606 6.78435 1.0
v 1.49012e-08 -14.7549 -5.13932 1.0
v -3.35276e-08 -16.4268 -7.10504 1.0
v 4.47035e-08 -17.7035 -8.19936 1.0
v 1.86265e-08 -13.8461 -9.19076 1.0
v -1.11759e-08 -15.4135 -6.13231 1.0
v -3.35276e-08 -13.1623 -12.438 1.0
v 4.47035e-08 -9.5203 -13.4744 1.0
v 1.49012e-08 -8.91593 -15.7128 1.0
v -2.04891e-08 -5.18348 -13.919 1.0
v -4.84288e-08 -4.70503 -14.8773 1.0
v 1.86265e-09 -3.56876 -14.2089 1.0
v 3.72529e-09 -3.13361 -14.2411 1.0
v -1.86265e-09 -2.3278 -14.225 1.0
v -1.30385e-08 -1.31246 -14.0155 1.0
v -2.6077e-08 -0.554995 -13.8382 1.0
v -2.6077e-08 0 -13.8382 1.0
v -1.11759e-08 0 -6.9 1.0
v -3.72529e-09 -2.8 -1.3 1.0
```

QUACK!!



Actually using a debugger
(A tool that is literally made for this purpose)

We will specifically talk about gdb(GNU debugger)



What can gdb do?

- stop our program at desired places and let us inspect it
- change the programs behaviour
- rewind our program
- give us information about various things e.g. stack frames and registers
- inspect coredumps
- and a few other things

How to debug a program with gdb

- Compile it with -g flag
- give the executable to gdb as and argument in command line

```
yusuf@vostro-5481:~/Documents/wsh_prep/gdb/example-code$ make hello
gcc -g    hello.c   -o hello
yusuf@vostro-5481:~/Documents/wsh_prep/gdb/example-code$ gdb hello
GNU gdb (GDB) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hello...
(gdb)
```

Breakpoints and watchpoints

- breakpoints stop our program when a certain point is reached
- watchpoints stop it when some value/expression changes
- watchpoints may slow your program heavily whereas breakpoints usually don't

How are they implemented?

- both breakpoints and watchpoints can be implemented in hardware
- if not debugger developers have to do the dirty work

layout src and list commands

```
yusuf@vostro-5481: ~/repositories/cclub-gdb-workshop/examples 191x46
hello.c
        4  {
B+>     5      printf("hello gdb\n");
        6
        7      return 0;
        8  }
        9
       10
       11
       12
       13
       14
       15
       16
       17
       18
       19
       20
       21
       22
       23
       24
       25
       26
       27
       28
       29
       30
       31
multi-thre Thread 0x7ffff7f966 In: main                                    L5    PC: 0x55555555513d
(gdb) info locals
No locals.
(gdb) p "Hello"
$1 = "Hello"
(gdb)
```

layout src and list commands

- **list** commands show a portion of the source code
- **layout src** splits screen into two allowing us to view the source code while typing gdb commands
- we can type **tui disable** to get out of text user interface

Inspecting function call stack

- where/backtrace/bt
- where fullc
- frame FRAME_NUM
- info locals
- info args

Where to get help?

- help command
- https://sourceware.org/gdb/onlinedocs/gdb/
- https://www.sourceware.org/gdb/documentation/

Rewinding

Altering execution

Following commands can be used to alter the program execution

- set var
- call
- compile code
- jump

defining your own gdb commands

When gdb is not enough

Posted by u/[illegible] 2 years ago

7

std::unordered_set::find() causes segfault

UPDATED

So I have this function that generates a unique id for a game object:

```
inline unsigned short generateId() {

        int i = 0; for(; ids.find(i) != ids.end(); i++) {} ids.insert(i); return i;

}
```

where `ids` is an `std::unordered_set<unsigned short>`. What it basically does is it finds the smallest available id and returns it. But when I call `ids.find(i)` it throws a segfault. Here's what gdb says:

```
Thread 1 received signal SIGSEGV, Segmentation fault.

0x00408370 in std::\_Hashtable<unsigned short, unsigned short, std::allocator<unsigne

at C:/mingw32/lib/gcc/i686-w64-mingw32/8.1.0/include/c++/bits/hashtable.h:643

643           { return \_\_hash\_code\_base::\_M\_bucket\_index(\_\_k, \_\_c, \_M\_bu

(gdb) info stack
```