Approximate Methods in Geometry

Practical Part

Jens K. Mueller

jkm@informatik.uni-jena.de

Department of Mathematics and Computer Science
Friedrich-Schiller-University Jena

Friday 10th December, 2010

Today's lecture: Debugging

Debugging

Debugging is about fixing bugs.

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

Brian Kernighan

Preventing Bugs

- Enable warnings all the time
- Fail early (assertions, contracts)
- Testing
- Logging
- Automatic Checking (e.g. for Memory leaks)

You have a bug!

- Record the bug
- Make the bug reproducible
- Contrast with (last) working version
- Examine the program
- Use tools
- Keep a History

Examining your Program

- Reason about your program
- Get a stack trace
- Enable additional output
- Divide and Conquer
- Numbers/Visualize

Still No Idea

- Explain your code to some else
- Step through the program

Fixing

- Make sure it's a proper fix
- Fixing one bug might need fixing other as well
- Make sure all test pass
- Update history

GNU Debugger

Run program under GDB

```
$ gdb -args ./program arg1 arg2
```

and then r(un)

Run program and attach GDB to it

```
$ ./program arg1 arg2 &; gdb program $!
```

and then c(ontinue)

Time traveling with GDB

How to debug your program reversely

1. Set a breakpoint
2. Run
3. Record
4. Continue

Use reverse-{step,continue} etc.