

Debugging

Kinds of Errors

- *syntax errors* mean that the program code is invalid
 - the program won't run – the compiler gives a line number and error message
 - see Appendix A for help with common syntax errors
- *bugs* are problems with the program logic that show up at runtime
 - the program runs but crashes – *runtime errors* stem from particular values leading to an invalid state
 - see Appendix A for help with some common runtime errors
 - the program runs without crashing but doesn't do what it is supposed to

Debugging

Adm. Grace Hopper, 1906-1992
computer scientist, mathematician, US
Navy rear admiral
known for: a pioneer of computer programming
– high-level English-like languages



9/2

9/9

0800 Antan started
1000 " stopped - antan ✓
1300 (030) MP-MS 1.52776915 (2) 4.61592509 (-2)
030 PROZ 2.13047645
030 " 2.13067645
Relays 6-2 on 030 failed speed speed test
in Testory
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
1600 antan started.
1700 closed down.

1.2700 9.030 847 025
9.037 896 985 correct

Relay 3145
Relay 3370

First actual case of bug being found.

first actual
bug, 1940s

Locating and Diagnosing Bugs

Tools –

- incremental development
 - if you add only a small piece at a time and test as you go, any new problem is likely to be in the part just added
- comment out sections that don't seem relevant
 - if the problem goes away, the most recently commented-out section is potentially the culprit
- remember that the program does exactly what you tell it
 - observe what is wrong in the sketch and find the part of the program where you said to do that
- use `println()` to trace what the program is actually doing
 - print values of variables to identify places where they don't match what you expect
 - print messages to trace the flow of control – does the computer even get to a certain point?

Locating and Diagnosing Bugs

Strategy –

- identify possible explanations for what is going wrong, and take steps to rule each possibility in or out
 - start with the most basic explanation
 - be careful to find the root cause

Example

There's supposed to be a red square, but there isn't one.

- there isn't code to draw the red square
 - locate the line(s) in the sketch where the red square is drawn

```
fill(255,0,0);  
rect(x,y+100,w,w);
```

- there is code to draw the red square but it isn't being executed
 - use `println` to print something immediately before that line – do you see that output?

```
fill(255,0,0);  
println("drawing square!");  
rect(x,y+100,w,w);
```

Example, continued

- the red square is being drawn, but something else is being drawn on top of the red square
 - comment out everything after the line(s) where the red square is drawn
- the red square is being drawn, but outside the window or with width or height 0
 - use `println` to print out the variables involved

```
fill(255,0,0);  
println("drawing square! "+x+" "+(y+100)+" "+w);  
rect(x,y+100,w,w);
```

Example, continued

→ the x coordinate of the square is incorrect

```
fill(255,0,0);  
println("drawing square! "+x+" "+(y+100)+" "+w);  
rect(x,y+100,w,w);
```

- find the root cause
 - is x meant to be the square's position?
 - check comments
 - check `rectMode` – CENTER vs CORNER
 - if so, the problem is x's value
 - trace backwards from where the square is drawn – where does x's value come from?
 - if not, the problem is using (just) x as the square's position