Data Structures in C

Prof. Georg Feil

Debugging in Dev-C++

Summer 2018

Acknowledgement

- These lecture slides are based in part on slides by Professor Magdin Stoica
- Additional sources are cited separately

Types of Errors (bugs)

- There are two major types of errors:
  - Syntax errors
  - Runtime errors
- Syntax errors
  - Are caught by the compiler (or linker), some may be warnings
  - Are almost always easier to fix than runtime errors
  - … read the error message (start with the first one)
- Runtime (logic) errors
  - Discovered while testing, or “using”, the program
  - Can be much harder to tell why the error/bug is happening
  - You can try adding some print statements (remove them later)

Much more powerful way to debug: Use the debugger!

Dev-C++ debugging

C:\cygwin\home\feilg\Teaching\Sheridan\201505\PROG20016\srcb\dog.c - [Debugging] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project | Classes | Debug

weight = 0
count = 1

dog.c

```
18
19  int main(int argc, char** argv)
20  {
21      float weight = 0;
22      int count = 0;
23
24      while (true)   // Loop until 'break'
25      {
26          // Prompt for input and read the weight in pounds
27          printf("Enter the weight of dog %d: ", ++count);
28          int parms = scanf("%f", &weight);
29          if (parms != 1) {
30              printf("Bad input, please try again\n");
31              fflush(stdin);
32              continue;
33          }
34
35          if (weight == 0)
36              break;  // Quit if weight is 0
37
38          bark(count, weight);
39          printf("\n");          // Leave a blank line
40      }
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Debug | Add watch | Next line | Continue | Next instruction

Stop Execution | View CPU window | Into function | Skip function | Into instruction

Send command to GDB: next

->->breakpoint 3

Breakpoint 3,
->->frame-begin 0 0x4015cd

me-function-name

me-args

Evaluate:

Line: 1 Col: 19 Sel: 0 Lines: 44 I Insert Done parsing in 0.047 seconds

"Watch" variables

Debug start

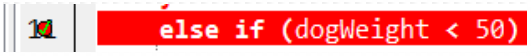
Stopped at this line (blue)

Breakpoint (red)

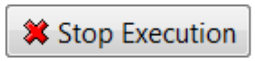
Debug control buttons

Print values or expressions

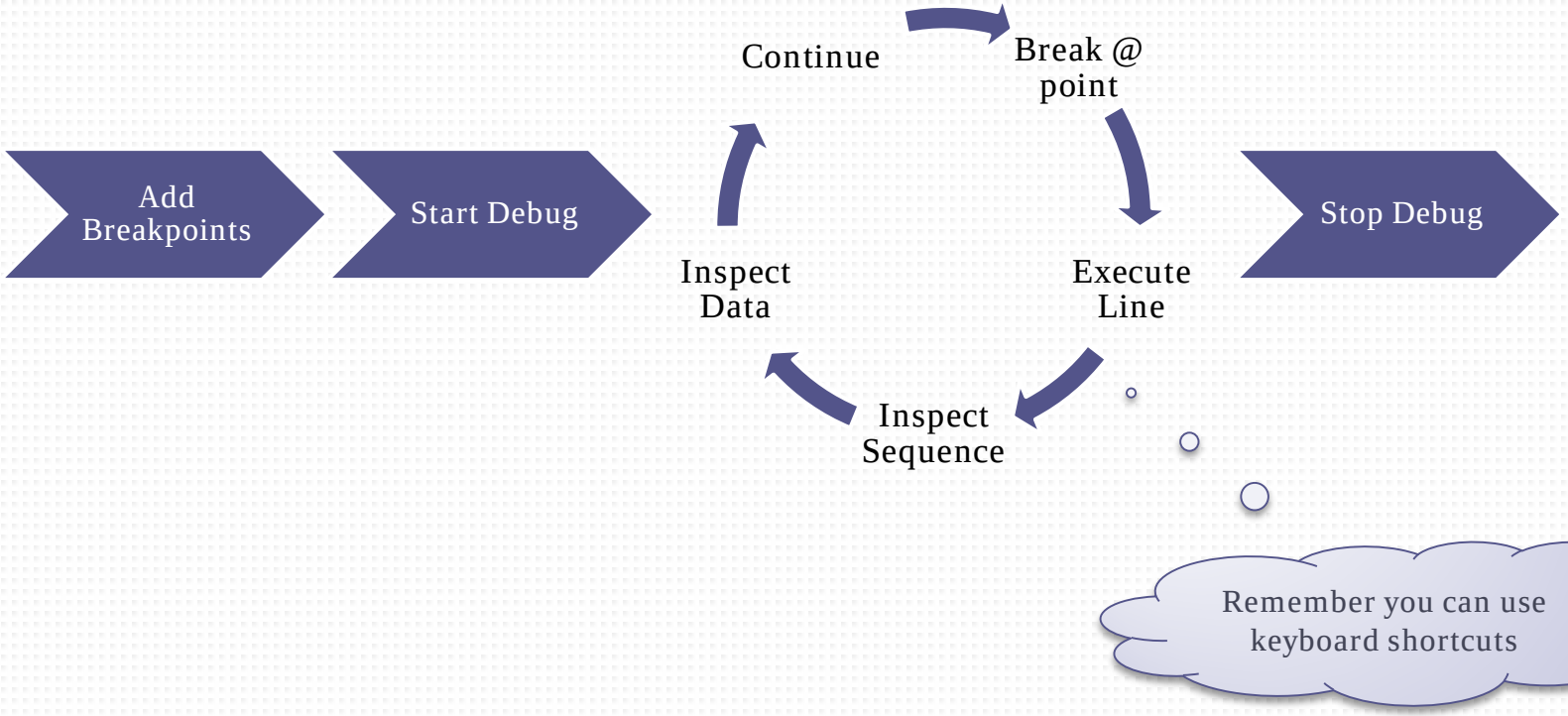
Debugging Workflow

- Add breakpoints in the areas of interest 

  - Breakpoints tell the debugger to stop at a particular line of code
  - Click on the far left of the statement (line number)
- Run the program using the debug button 
- Dev-C++ will open the debug window
- When the program stops at a breakpoint single-step interactively to slow it down using the debug control buttons

- Inspect values of variables using “Add watch” or “Evaluate”
- When done click stop to terminate the program 

Debugging Workflow

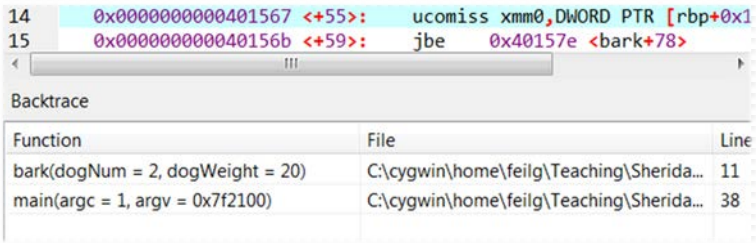


Single-step Debugging

- (Step) Into Function (F8)
  - Single-steps to the next line of code, including going inside your functions (to debug functions)
- Next Line (F7)
  - Single-steps to the next line of code, without stopping inside functions called (use this to skip over functions you don't need to debug)
- Skip Function
  - Runs until the function you're in returns. The program stops at the line following the function call that just ended

View Stack Trace

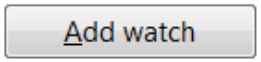
- The stack trace shows you the sequence of nested function calls
  - Shows the big picture of "where you are" in the program
- Use the "View CPU Window" button to show the stack trace (called backtrace in Dev-C++)
  - The function at the top of the the list was called most recently
  - The next function in the list called the previous one, and so on

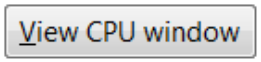


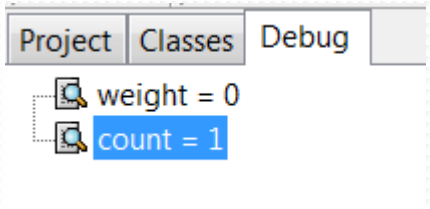
- Also shows assembler instructions and CPU registers!

Debugging Commands Summary

- Set/clear breakpoint: Click on far left of statement, or F4 
- Start debugging: F5 
- Single step: F7 (next line/step over), F8 (into function/step)
- Continue running (resume): Continue button 

- Stop debugging: F6 
- Watch a variable (or calculation): Add watch button 

- Remove a watched variable: Click on item, press Delete
- View stack trace 



Exercise: “Debug” In Practice

- Check that debugging is enabled
  - Tools > Compiler Options > Settings tab > Linker tab > Generate debugging info (-g3) → Yes
- Open your program from Exercise 1 in the Functions slides (barking dogs)
- Set a breakpoint, compile, then run using the Debug button
- Practice to learn debugging
  - Breakpoints
  - Controlling the speed
  - Stepping over, in and out of functions
  - Check the value of variables using “Add Watch”
  - While single-stepping inside the bark function, use the “View CPU Window” button to show the stack trace