Debugging using Visual C++

Chuong Nguyen

Main Topics

- Why do I care?
- Development environment
  - Editor
  - Compiler
  - Debugger
  - IDE
- Debugging
  - Locate a bug
  - Find the cause
  - Fix the bug
- Debugging in action

Why do I care?

- Why don't you?
  - Write code → Compile → Test → Something wrong → Debug
- Understanding the concepts is critical. Also the "how to" helps you save hours or days (you can use that time to sleep, instead).
- This is a general debugging tutorial, using Visual C++ as an example

Development Environment

- Text Editor
  - Simple application that lets you create raw (unformatted) documents, NO fancy features (NO bullet points, NO underlining, etc.)
  - e.g. Notepad, Textpad | emacs, xemacs, vi, pico
- Compiler
  - Complex application that converts your source code to machine language
  - e.g. Microsoft C/C++ compiler | gcc, g++
- Debugger
  - Complex application that lets you walk through the execution of your program
  - e.g. Microsoft Visual Studio debugger | gdb
- Integrated Development Environment (IDE)
  - An IDE = Editor + Compiler + Debugger + other fancy features
  - Visual C++ is an IDE

Overview of Debugging

- Bug = error
- Debug = try to fix the error

- Locate the bug
  - Narrow down which lines of code introduce the bug

- Find the cause
  - Understand why those lines of code do the wrong thing

- Fix the bug
  - Now what, how are you going to fix it?

- In this tutorial, we focus on how to use the Debugger to “Locate the bug”

Visual C++ debugger

- Start vs. Start without Debugging
- Debugging techniques
  - Breakpoints
  - Trace execution (walk through): single stepping
  - Monitor variables and function calls
    - Watch variables
    - Function call stack

Start vs. Start without Debugging

- Start without Debugging : Ctrl+F5
  - After execution, the console window is still there and you can see the results
  - However, it's not what we want for this tutorial since it won't stop at our breakpoints for us to examine the program

- Start (with Debugging): F5
  - The program will stop at the breakpoints so we can take a deeper look and debug it

Breakpoints

- Why we need them
  - To show the machine world that we have control over them
  - To stop the program at any line of code we want

- How to set/unset them
  - Click on the gray margin on the left side of the line of code
  - Unset: do the same thing

Trace (walk through) your program’s execution

- **Single stepping:**
  - Run at a time one piece of your code (one line of code or a function)
- **Step in: F11**
  - Step into a function
  - Pause at the beginning of the function
- **Step over: F10**
  - Execute a line of code no matter what it is (assignment, comparison, cout, cin, functions…)
  - Pause at the next line of code
- **Step out: Shift+F11**
  - Get out of the function that I’m in right now (execute the rest of the function but don’t show me what’s going on)
  - Pause at the next line of code right after the function call.

Monitor variables and function calls

- **Debugger windows**:
  - Only available when you're in debugging mode.
  - Useful windows to keep track of your program
    - Breakpoints, variables, function calls
- **Watch window:**
  - Keep track of your variables' values
  - Short-hand calculator (evaluate variables, expressions)
- **Call Stack window:**
  - Keep track of function calls (who called me).
  - Useful with recursive calls

Recap

- IDE = editor + compiler + debugger + other fancy features
- Write code → Compile → Test → Something wrong → Debug
- Debugger: lets you trace/walk through the execution of your program
  - Locate the bug
  - Find the cause
  - Fix the bug
- Debugger techniques:
  - Set break points: stop the program at a line of code
  - Trace execution: single step through your program
  - Monitor variables: look at values of the variables

Debugging techniques and Tips

- Don't duplicate code. Put repeated/reusable code in a function and call the function.
- To locate your bug, use binary search: repeatedly divide the search interval in half (put breakpoint at the middle, then at the middle of first half or second half, etc).
- Display line number next to line in the text editor in Visual C++
  - Tools/ Options/ Text editor/ C++
- C++ help:
  - Google: C++ FAQ Lite