

# **Keynote:** **Steps to Wisdom for C++ Developers**

Kate Gregory

# Steps to Wisdom for C++ Developers

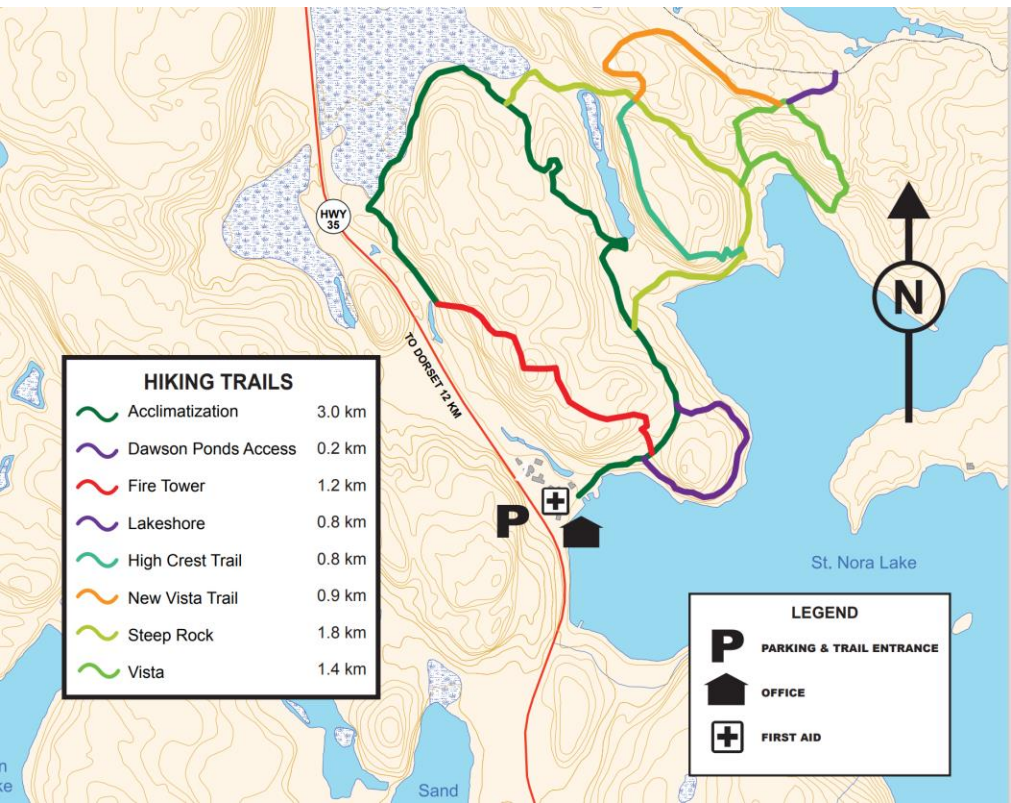
**Kate Gregory**

kate@gregcons.com  
@gregcons on Twitter

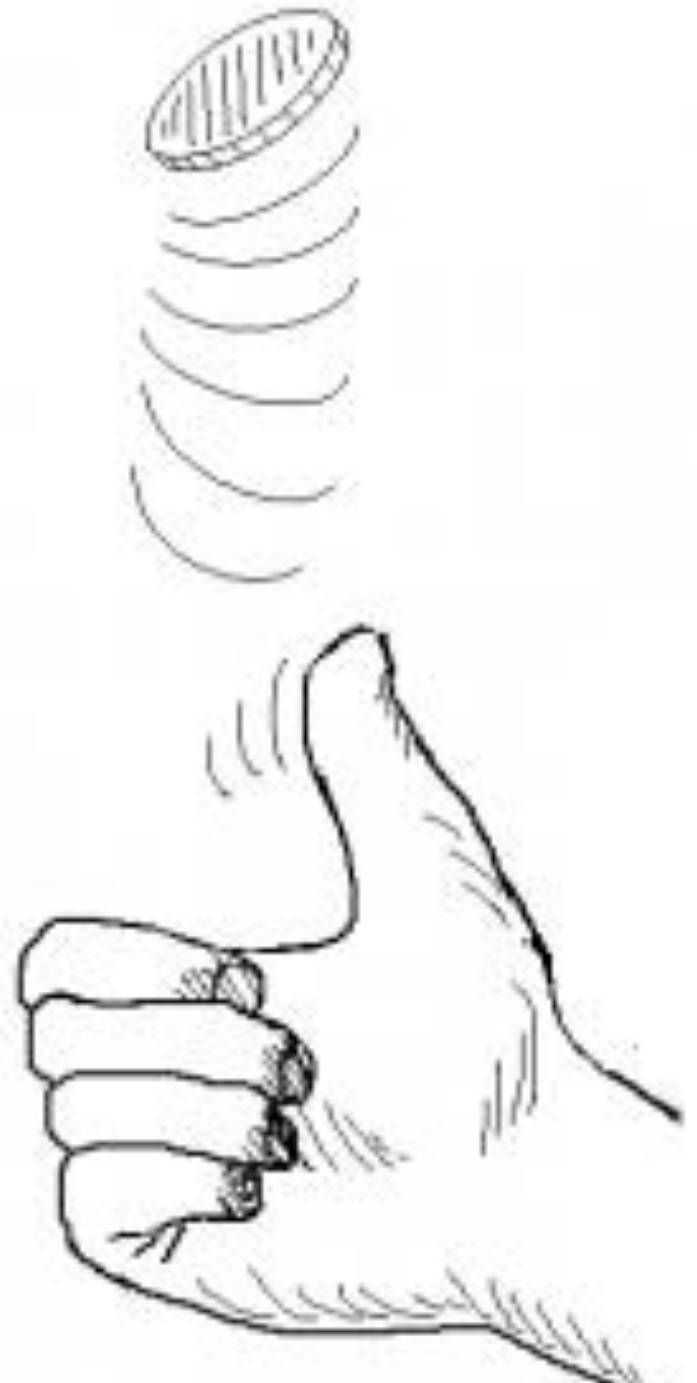












# What is Wisdom?

- Heuristics and guidelines that lead to better decisions
  - Easier or faster decisions
  - Better outcomes
  - Less worry afterwards
- Who defines better?
  - You
- Touchstones and metaphors that help you keep going
  - Improve your mood
  - Get through adversity

# Lessons I've Learned

- By messing up and realizing what would have been better
  - You can learn and discover wisdom like this over a lifetime
  - Or live the same year of experience over and over again
- By lucking into doing the right thing
  - And noticing it was right
  - And remembering a general rule from it
  - When things work, we often don't notice why it worked or what we did that was special
  - An easy life tends not to produce true wisdom
    - Born on third base, thinks he hit a triple
- By trusting a person who told me a handy rule or technique

# These Aren't Secrets

- You've probably heard some of these before
- Why don't you follow them, or do what they suggest?
- Can something simple and obvious really make a difference?
  - Yes
- Try trusting and see what happens



# These Aren't Everything

- I am surely missing more than I am including
- Some of this might not apply to you
  - Might even be completely wrong for you
- Take what works, and add more of your own
- Develop a habit of accumulating wisdom deliberately





# This Should Work

# This Really Should Just Work

- “Try it now”
- “If all else fails, read the instructions”
  - Or the error message
- You’re not an imposter, you’re a learner
- Come at it another way, try another approach



# You Have to Actually Do the Work

# Do the Work

- Take a deep breath and do it
- Put in the time
- Keep trying
- There's no quick way to become an overnight success
- It's amazing how long a thing takes to finish when you're not working on it
- If it was easy, anyone could do it



# Somebody Should

- Pointing out that something needs to be done is an important skill
- Actually doing things that need to be done is a different skill
- If you can't do it now, can you make it get done?
  - Add an issue
  - Get it on a list of tasks or objectives
  - Offer to help the person who would do it
- You may end up doing it eventually

# Make sure it's the right work

- You can't push a rope
- Don't just put in time on tasks that don't accomplish anything
- My mantras:
  - Do a lot
  - Do the right things
  - Do them well
  - Stop when you're done.

# Observe and Remember



# Take Notes in Meetings

- The act of taking notes helps memory
  - Especially when you write them by hand with a pen
- Pay attention! Stay zoned in whenever you're interacting with others
  - Miss nothing
- Having notes to search later makes you the go-to for history, lore, rulings, and decisions
  - What you choose to write down becomes more important if no-one else records anything
- Reminding people “we already settled this” doesn't just save time, it saves feelings
- Note-taking can be “office housework”
  - often put on historically-excluded people
  - don't sit back and take notes while “the important people” talk
  - use it as a source of power; write down the decisions you help to make

# Take Notes as you Work Alone

- Keep track of details you might need later
  - When did you start and with what config/settings?
  - What happened and when?
    - Expected
    - Unexpected
  - Exactly what the error was
  - How long things took
- Gives you the information you need when it's time to fix it

# Science Notebooks

- Standard issue for engineers in the 80s
- Bound – don't add or remove pages
- Use pen
- Date every page, every entry
  - Sometimes times too



# Running Notes

- For a spike or bug investigation
- For the first time you do something that you plan to do a lot
- Stream of consciousness
  - Screenshots
  - Pastes from emails or program output
  - Include dead ends, stuff that doesn't work out, things that turned out to be wrong
  - Pastes of bits of code
- Great place to put dead code you're removing
  - And why!
  - Can find it more quickly if you need to put it back
- Really helpful for things you have to set aside and pick up again after a gap
- Later, extract simple 7-step process or whatever
  - These are raw notes to builds things from if you turn out to need them

# Using Your Notes

- On the day you made them, because you need those details
- To find credentials (or where they're kept), paths, documentation links
- To write summaries and instructions
- To see how it used to work/look
- To back up assertions about how long things took
- To back up assertions about who wrote or created something
- To show, with confidence, how something happened
  - Possibly with a lawyer present

# Connections and Patterns




# Connections Between People

- Are your coworkers in your alliance?
- Peer support and recommendations require trust
  - Asking for advice or assistance
  - Being part of a community that will help without judging or mocking
  - [includecpp.org/discord/](https://includecpp.org/discord/) among others
- Friendships make life richer
- Sometimes things you do primarily to support someone else are very rewarding

# Connections Between People

**pacific++** Sydney, Australia | October 2018



**Generic Programming\***

David R. Musser<sup>1</sup>  
Rensselaer Polytechnic Institute  
Computer Science Department  
Amos Eaton Hall  
Troy, New York 12180

Alexander A. Stepanov  
Hewlett Packard Laboratories  
Software Technology Laboratory  
Post Office Box 10150  
Palo Alto, California 94303-0909



**Abstract**

Generic programming centers around the idea of abstracting from concrete, of finding algorithms to obtain generic algorithms that can be combined with efficient data representations to produce a wide variety of useful software. For example, a class of generic sorting algorithms can be defined which work with finite sequences but which can be instantiated in different ways to produce algorithms working on arrays or linked lists.

Four kinds of abstraction—data, algorithmic, structural, and representational—are discussed, with examples of their use in building an Ada library of software components. The main topic discussed is generic algorithms and an approach to their formal specification and verification, with illustration in terms of a partitioning algorithm such as is used in the quicksort algorithm. It is argued that generically programmed software-component libraries offer important advantages for achieving software productivity and reliability.

\*This paper was presented at the First International Joint Conference of ISSAC 88 and SASECC 88, Rome, Italy, July 4-8, 1988. (ISSAC stands for International Symposium on Symbolic and Algebraic Computation and SASECC for Applied Algebra, Algebraic Algorithms, and Error Correcting Codes). It was published in Lecture Notes in Computer Science 326, Springer Verlag, 1989, pp. 12-25.

The first author's work was sponsored in part through a subcontract from Computational Logic, Inc., which was sponsored in turn by the Defense Advanced Research Projects Agency (ARPA order 855). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the U.S. Government, or Computational Logic, Inc.



**SEAN PARENT**  
Generic Programming

© 2018 Adobe. All Rights Reserved.

3:17 / 1:19:56 • Generic Programming >

Pacific++ 2018: Sean Parent "Generic Programming"

# Connections Between Incidents

- Patterns
- Things that recur (and how it was handled before)
- This is just like x, except with y
- These often occur to you when you're not really thinking about them
  - Pay attention when that happens and write it down somewhere



# But Don't Overgeneralize

- Not all four-legged creatures are dogs
- Some things feel like they're about you, but aren't
  - People have other things in their life and are reacting to those
  - Rude job interviewers don't know you well enough to dislike to disrespect you, they're just being rude
- It was a memory leak last time, that doesn't mean it is this time
- It's always a good idea to check even when you're sure
  - Is it plugged in?
  - Check for agreement / consent

# Ask the Right Questions

# Good Questions

- “Some days I just want to... “
  - “And what would be different if you did?”
- Why hasn’t that happened yet?
- What do you think about this?
- Who is supposed to make sure that happens?
  - Not the same as “do that”
- What did you think that would fix?
- What’s the worst that could happen?
- What needs to change to prevent that?

# Good Questions

- Is this costing us money? How much?
  - Caution, delay, downtime, error, ...
- What is keeping this from moving forward?
- How would I find out?
- “Are you ok?”

# Ask Yourself Too

- These questions bring out hidden knowledge
  - One on one
  - In meetings
  - In background documents
- But they work on yourself
  - Interview yourself, ask some of these
  - Answer aloud
  - Prepare to be surprised



# Know Your Strengths

# Your Strengths

- What are you really good at?
- What do you really enjoy?
- When something is “in your wheelhouse” recognize it
  - Speak up
- Tell people what you are best at
- Know the value you bring
- Accept tasks with enthusiasm and confidence

# Your Weaknesses

- You don't have to tell people, but you need to know
- You can work on them if you like
- Or you can avoid those tasks
  - May require a different job
- Decline with grace when you can
  - “That’s not really one of my strengths”
  - “I think I’d be better placed doing [different thing]”
- Ask for help and support when you can’t

# Have Goals

# Guiding Star

- Have a vision, a mission, a guiding star
- More likely to reach it, or at least get near it
- Makes decisions easier
  - A or B?
  - Which one moves me (or this project or this team) towards the goal better?
  - If you don't immediately do that, why not?
    - What else is competing with the goal? Are there two goals?
    - Dig into what the tradeoff or issue is
    - What scares you?

# Goals

- Set your priorities
- Know what to ignore
- Make choices
  - Don't write "high perf" all the time
  - Don't avoid parts of the language all the time
  - What does this project need? What are your goals for this piece of software?
- Feel more comfortable with the choices you've made
- Remember your goals
  - Keep them front and centre
  - Use them throughout your work process



# Your Goals

- Not someone else's



**Adam Grant** ✓

@AdamMGrant



The saddest form of success is realizing that the goals you achieved weren't yours at all.

In the short run, pursuing other people's dreams earns approval. In the long run, it's a recipe for regret.

A meaningful purpose doesn't maximize your status. It matches your values.

11:06 AM · Jun 8, 2023 · **415.5K** Views

# Your Goals

- Not someone else's
- Write your own eulogy
- Examine those “conflicting goals moments”
  - They reveal goals you may not have been really aware of
  - You can consciously change your mission statement

# Look for the Learning

# Learning

- Find the joy in the work you do now
  - You'll be eager to learn new stuff every chance you get
- The only thing better than learning from your own mistakes is learning from someone else's
- Don't pay for the same lesson twice
- Everything is a learned skill. Everything.
- You're wiser than you think

# Catch Phrases Help

- Quick summaries to remind you of the larger and longer truths
  - Naming is Hard
  - That's a rotate!
  - Better safe than sorry
  - Don't pay for 90% of luxury
- Make up your own to remind you what matters to you



- This Should Work
- Do The Work
- Observe and Remember
- Make Connections
- Ask the Right Questions
- Know Your Strengths
- Have Goals (and Use Them)
- Look For the Learning

