CSCI 2270

Data Structures & Algorithms

Gabe Johnson

Lecture 21 Mar 6, 2013

Design Practice

Upcoming Homework Assignment

HW #6 **Due: Friday, Mar 8**

Priority Queues

We'll work on Priority Queues all day today in small groups. The PDF is up in the HW 6 folder.

Lecture Goals

1. Short Lecture on Design Process
2. Small Groups

Problem Framing & Solving

“A QUESTION WELL-ASKED IS HALF-ANSWERED.”

- Witty Saying Attributed to Charles Kettering

“A WITTY SAYING PROVES NOTHING.”

- Witty Saying Attributed to Voltaire

Problem Framing

Questions about the Question

Can’t we just not do this and call it a day?
Who cares?
Do *I* care?
What’s the question to begin with?

Problem Framing

Questions about the Problem

What is the problem?
Who has this problem?
Can't the problem be solved *this other way*?
Can't we just use a library function?

Game of 15

Take turns with your opponent, claiming numbers.
First to sum to 15 wins! Here's a game in progress:

1, ~~2~~, ~~3~~, ~~4~~, ~~5~~, 6, 7, 8, ~~9~~

X: 2 + 4 + 3 = 9 O: 9 + 5 = 14

Game of 15

2	7	6
9	5	1
4	3	8

You could frame the game like this.
Look familiar?

Game of 15

x		
o	o	
x	x	

It is O's turn.

X: 2 + 4 + 3 = 9 O: 9 + 5 = 14

Problem Framing

Now on to data structures. Even though this looks like a step-by-step, **be aware this is an artifact of writing things down. Design is iterative and nonlinear.**

<u>Questions about a Data Structure Design:</u>

What operations are needed?

What are the invariants/required?

What data are needed?

How are they related?

Problem Framing

In software (at least, low-level stuff like designing a data structure), framing a problem means understanding what the requirements are, what data we need, how they relate.

It is **not** about writing code. It is **not** about C++ syntax vs. Java syntax.

Problem Solving

Once you think you've framed the problem you can try to solve it. **Like I said, design is not linear. You will probably have to go back and forth between framing and solving several times.**

What particular data types do we have?
What files are needed? Header files? CPP files?
How do we build the thing?
How do we test the thing?

Problem Solving

Strategy 1:

Write a ton of code. Spend hours between compilations. Bite off huge chunks.

Strategy 2:

Write a few lines; compile. Fix errors. Look at print statements. Sanity check. Address problems. Repeat. Keep the problem in mind.

Today

People who know what's up: please self identify.

People who don't: also please self identify.

Remember, one of the smartest things you can say is "*I don't know*". It is hard to learn anything when you think you already know it.

We'll break into groups of 4 or 5 and design the priority queue. These things can be very lame, so if you feel lameness setting in, do something **strange**.

Useful Questions

- Can’t we just not do this and call it a day?
- Who cares?
- Do *I* care?
- What’s the question to begin with?
- What is the problem?
- Who has this problem?
- Can’t the problem be solved *this other way*?
- Can’t we just use a library function?
- What operations are needed?
- What are the invariants/required?
- What data are needed?
- How are they related?
- What particular data types do we have?
- What files are needed? Header files? CPP files?
- How do we build the thing?
- How do we test the thing?