Course Introduction

Dr. Mattox Beckman

Illinois Institute of Technology
Department of Computer Science

Objectives

Welcome to CS 331! The objectives for today's lecture are:

Organizational

- Know your instructor.
- Know how the class will be run, and why.
- Know how your grade will be determined.
- Know when the exams are.

Content

- Know the objectives for this course.
- Know why we should study data-structures.
- Learn about our class language.

Course Objectives

By the time the course is finished, you should be able to:

- Explain, implement, and apply the following data-structures:
  - lists (unordered and ordered),
  - stacks,
  - queues,
  - expression trees,
  - binary search trees,
  - heaps, and
  - hash tables.
- Analyze the time and space complexity of algorithms using asymptotic upper bounds (big-O notation).

More Course Objectives

But wait! There's more!

- Explain and use references and linked structures.
- Outline basic object-oriented design concepts: composition, inheritance, polymorphism.
- Outline basic functional programming design concepts: immutability, values, higher order functions.
- Write and test recursive procedures, and explain the run-time stack concept.
- Analyze searching and sorting algorithms, and explain their relationship to data-structures.
- Choose and implement appropriate data-structures to solve an application problem.
- Understand techniques of software development, such as unit testing and version control.

Me!

Name	Mattox Beckman
History	PhD, Fall 2003, University of Illinois at Urbana-Champaign
Research Areas	Programming Languages, Mathematical Foundations of Computer Science
Specialty	Partial Evaluation, Functional Programming
Professional Interests	Teaching; Partial Evaluation; Interpreters; Functional Programming; Semantics and Types; Category Theory
Personal Interests	Cooking; Go (Baduk, Wei-Qi, Igo); Theology; Evolution; Greek; Meditation; Kerbal Space Program; Home-brewing; ... and many many more ...

Teaching philosophy is available at
http://mccarthy.cs.iit.edu/mattox/static/teaching-philosophy.pdf

My Responsibilities

My job is to provide an “optimal learning environment”.

- Assignments will be clearly written and administered.
- Questions will be answered in a timely fashion.
- Objectives of lectures and assignments will be clearly communicated.
- Grades will be fair, meaningful, and reflect mastery of course material.
  - C grade means “can reliably recognize the correct answer”
  - A grade means “can reliably generate the correct answer”
- **If something’s not going the way it should, tell me!**

Your Responsibilities

- Check the course web page frequently.
  (currently http://mccarthy.cs.iit.edu/cs331)
- Do the lab assignments in order to learn them.
- Attend lectures — or find out what happened there.
- **Take responsibility and initiative in learning material** — experiment!

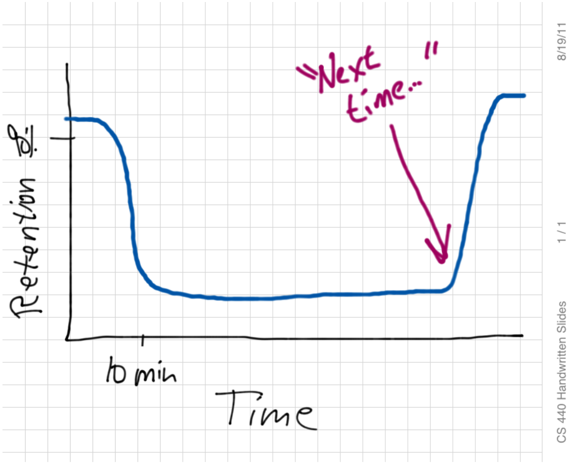
You are the one primarily responsible for your education.

Lectures

Speaking of lectures...

- The lecture is ancient technology; invented before the printing press.
- “Transforming lives, rehashing the past?”
- What usually happens during a lecture?

Attention vs. Time



Some observations about learning

- Traditional lectures are hard because:
  1. You have to be there at a certain time.
  2. ...and you have to be awake.
  3. ...and you can't "rewind" if you miss something.
  4. But at least you can ask questions! (If you're not shy.)
- Homeworks are hard because:
  1. What seemed obvious in lecture is not obvious later.
  2. You can't ask the professor for help until office hours (or until (if) they check their email).
  3. The one time you see the professor is during lecture, and then they are busy lecturing.
- Proposal: we're doing this backwards. Let's do it the right way instead.

Reverse Lectures

The Right Way

- Lectures will be screen-cast and made available on the course web site.
  - Usually 2–4 mini-lectures, about 10 minutes each.
  - Transcript will be available (probably).
  - Viewable on mobile devices.
  - Hard part: you do need to see them before the corresponding class period.
- During class:
  - Review time: “Any questions about the lectures?”
  - Activity. Work in groups of 2–3 people, reinforce lecture concepts, prepare you for exams. Worth 10% of your grade.
  - Homework. I don’t think we’ll have time for homework, but let me know if you’re bored and I’ll come up with something.
- This method is not common, but has been thoroughly tested, and **it works**.

Activities

- Best research indicates that students learn better when lectures include in-class activities.
  - Bring pen and paper.
  - You can talk about the example problems during the activity time, but this is not an invitation to talk about other things, or to talk during the rest of the lecture.
  - Answers will be included in the on-line slides, but not handed out.

Exams

- The purpose of an exam is to measure mastery of material.
- 2 midterm exams — worth 25% each.
  - They will be held during class.

    Dates Friday, February 27; Friday, April 3
- Final exam — worth 25%
  - Date: To be determined
  - **Determined by lecture schedule, not lab schedule!!!**
  - Cumulative
  - Nice British System

Labs

- Roughly 5 to 10 “For Credit” Lab Session — collectively worth 15%
- Labs are usually just small programming assignments.
- Purpose of labs is to reinforce material and give practical experience.
- The lab has a grading script. You can turn in your lab for grading as many times as you want (until it’s due).
- If you wait until the last hour to turn it in, you will likely be very very sorry.
- You are allowed to work together, and encouraged to bring your lab to class.

Grade Guarantees

The course will not be graded on a curve or by ranking. Instead, we have the following grade guarantees:

- 85% A
- 70% B
- 55% C
- 40% D

Language Choice

The Course Language is Clojure.

"The most fun I have ever had programming..."

- History
  - 1958: McCarthy creates the Lisp programming language at MIT.
  - 1962: First Lisp compiler written in Lisp.
  - 1972-ish: Sussman and Steele develop Scheme to study a programming concept called the actor.
  - 1975: Paper on Scheme published.
  - 2009: Rich Hickey releases Clojure.
- After our introductory slides I will introduce Clojure more thoroughly.
- Don't worry—I'm assuming that nobody here has seen this language before!

Tool: Unit Testing

- We will make extensive use of unit testing in this course.
- Look up “`midje`” if you want a head start.
- Your lab grades will depend more on the tests than on the code!
  - Your Code vs My Tests
  - Your Tests vs My Solution — Qualifying Run
  - Your Tests vs All Kinds of Broken Solutions

Tool: Linux

- GUI goal: be pretty. It's for people who don't actually like computers.
- CLI goal: be efficient.
  - Secret advantage: it's *scriptable*.
- There will be a video about how to use Linux; you really only need to know a few things.
- Expect a steep learning curve, but it will be worth it.

Tool: Git

- Git is a Source Code Management system.
- Used by linux kernel, X.org, OLPC, etc.
- I will give a presentation on this for your labs.
- Expect a steep learning curve, but it will be worth it.

Tool: Editor

- You should pick one. Notepad doesn't count.
- Recommendations:
  - LightTable
  - emacs
  - vim
  - emacs + evil plugin
  - eclipse + counterclockwise
- Expect a steep learning curve, but it will be worth it.

Starting CLOJURE

- Editor (Emacs): `cider-jack-in`
- LightTable
- Command line: `lein repl`

I usually use `emacs` with `vim` keybindings for my programming.

Next time...!

Here are the features we will cover next time: If there's time, we will look at some of it today.

1. Numerics
2. Operations
3. Defining Variables
4. Defining Functions
5. Local Variables
6. If / Conditionals
7. Special things: symbols, lists
8. Functions (lambda)

But First!

- Imagine that your client is a DJ, who does a lot of freelance work to a diverse clientele.
- He has a huge collection of MP3 files. All legal, of course.
- He wants you to write a database to keep track of them.
  - What properties of the music will you keep track of?
  - For the front end, what kinds of operations do you think you would write? "What kinds of questions will you want to ask?"
- Work in groups of 2–3 to build a consensus.