

## **Class 02: An Introduction to CSC151**

**Held:** Tuesday, January 23, 2007

**Summary:** Today we begin to consider the structure and content of the course. We also prepare ourselves to use the MathLAN workstations.

### **Related Pages:**

- EBoard.
- Lab: Getting Started in the MathLAN.
- Reading: The Linux Environment in the MathLAN.

### **Due**

- HW 1

### **Notes:**

- EC: Thursday, 11 a.m., Johanna Meehan on vocation (or lack thereof).
- EC: Thursday, 4:15 p.m., John David Stone on the game of life.
- Reading for Wednesday: The DrScheme programming environment.
- I have started to respond to your questions. When appropriate, I've also posted my responses to a Web page.

### **Overview:**

- Lessons from PB and J.
- Common parts of an algorithm.
- About the course.
- Lab.
- Some administrative details.

## **Reflections on the PB and J Problem**

*Left for you to fill in the details.*

## **The Parts of an Algorithm**

- As you may have noted, there are some common aspects to algorithms. That is, there are techniques that we use in many of the algorithms we write.
- It is worthwhile to think about these algorithm parts because we can rely on them when we write new algorithms.

## Variables: Named Values

- As we write algorithms, we like to name things.
- Sometimes we use long names, such as “the piece of bread in your dominant hand”.
- Sometimes, we use shorter names, such as “bread-dom”.
- As we start to write more formal algorithms, we will need techniques for noting which names we are using and indicating what they name (and, sometimes, what kind of thing they name).
- We call these named values *variables*, even though they don’t always vary.
- We need to be careful to use unambiguous names. Recall the problems with using “it” in your descriptions.

## Parameters: Named Inputs

- Many algorithms take data as input (and generate other data as output).
- Our PBJ algorithm takes bread, jelly, and such as input.
- A “find square root” algorithm would take a number as input.
- A “look up a telephone number” algorithm might take a phone book and a name to look for as inputs.
- In each case, the algorithm should work on many different inputs.
- The algorithm works as long as the input is “reasonable” (we can’t find the square root of a piece of bread and we can’t make a PBJ sandwich with tunafish).
- We call these inputs *parameters*.

## Conditionals: Handling Different Situations

- At times, our algorithms have to account for different conditions, doing different things depending on those conditions.
- In our PBJ algorithm, we might check whether the jar of peanut butter is open or what kind of lid is on the jelly jar. We call such operations *conditionals*.
- Conditionals typically take either the form  
if *some condition holds* then *do something*
- Here’s a slightly more complex form  
if *some condition holds* then *do something* otherwise *do something else*
- At times, we need to decide between more than two possibilities. Typically, we organize those as a sequence of tests (called *guards*) and corresponding things to do.

## Repetition

- At times, our algorithms require us to do something again and again.
- In our PBJ algorithm, we may have had to turn the twisty-tie again and again until it was untwisted.
- We call this technique “*repetition*”.
- Again, repetition takes many forms.
- We might do work until we’ve reached a desired state.
- We might continue work as long as we’re in some state.
- We might repeat an action a fixed number of times.

- You can probably think of many other forms of repetition.

## Subroutines: Named Helper Algorithms

- Many algorithms require common actions for their operation.
- For example, to make N sandwiches, you benefit from knowing how to make one sandwich.
- To make a peanut butter and jelly sandwich, it helps to know how to spread something on bread.
- We can write additional algorithms for these common actions and use them as part of our broader algorithm.
- We can also use them in other algorithms.
- We call these helper algorithms “*subroutines*”.

## About This Class

- Computer Science 151 has a number of goals
  - To introduce you to fundamental ideas of computer science: abstraction, algorithms, and data
  - To enhance your problem-solving skills and give you experience in formal representation of problems and solutions.
  - To introduce you to two primary paradigms of problem solving: functional and imperative.
  - To give you some programming skills that you can apply to problems in other disciplines.
- I expect and hope that you will find CSC151 different from any class you’ve taken in the past.
  - We use a different format than many classes: a collaborative, workshop-style format. (You may have seen this format in other introductory science courses; we do it somewhat differently.)
  - Computers and computer science also require you to think differently. I expect that you’ll exercise some brain cells you may have forgotten you have. (And after all, isn’t liberal arts education an exercise in thinking in as many ways as you can?)
- Like most computer science courses, CSC151 will have both theoretical and practical components. I hope you will enjoy relating the two.

## Lab: Getting Started in the MathLAN

- We’ll break about midway through today’s class to get you set up working in the MathLAN.
- This “lab prep” is really pointless and annoying, but also necessary.
- Do the lab.

## Administrative Issues

- *I am not sure whether or not I will cover these topics in class. They are included for your edification.*
- Please refer to the course web site for more details.
- Teaching philosophy: I support your learning.
- Policies
  - Attendance: I expect you to attend every class. Let me know when you’ll miss class and why.
  - Grading: I’m a hard grader. I don’t grade everything.
  - Course web.

- Etc.
  - Daily work
    - Attend class, work on lab and participate in discussion.
    - Finish the lab in the evening.
    - Do the reading for the next class in the evening.
  - The exams
    - Three take-home exams during the semester. Plan to spend six to eight hours on each one.
    - An *optional* final to make up for a bad exam grade.
    - Take all three exams anyway.
  - The labs
    - Available online.
    - Being re-written as the semester progresses.
    - I'll require more formal writeups of a few labs a semester.
  - The homework
    - About twenty.
    - One makeup at the end of the semester.
  - Academic honesty
    - Core to the academic process.
    - My basic policy: Don't cheat.
    - The college's basic policy: Cite carefully.
    - Significant breakdowns in CSC151 last semester.
    - Read my handout on academic honesty carefully.
- 

Copyright © 2007 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.