Class 02: An Introduction to CSC151

Held: Monday, 31 August 2009

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

Related Pages:

- EBoard.
- Lab: Getting Started with Linux.
- Reading: Grinnell's Linux Environment.
- Due: Assignment 1: Class Basics.

Notes:

- I have started reading through your answers to the introductory survey, but have not gotten all the way through. Expect answers to questions tomorrow.
- To read for tomorrow's class: The GNU Image Manipulation Program.

Overview:

- Drawing smiley faces, revisited.
- Lessons from day one.
- Common parts of an algorithm.
- About the course.
- Getting started with Linux.
- Some administrative details.

Drawing Smileys, Continued

Sam will attempt to follow some instructions for drawing smiley faces.

Reflections on Drawing Images

What, if anything, did you learn from the drawing exercise?

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).
- Your smiley-face algorithms might take the size of the face (or board) 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 lok for as inputs.
- In each case, the algorithm should works on many different inputs.
- The algorithm works as long as the input is “reasonable” (we can’t find the square root of a smiley face and we can’t draw a circle with pi).
- 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 smiley-face algorithm, we might check what kind of face we want to draw, or whether the marker is open or closed. 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 the turtle drawing, we had to repeatedly draw a small line and turn a small amount.
- 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, the circle group relied on a "draw a circle" routine.
- The pixel group could immediately use the circle group's solution by simply writing their own draw a circle routine".
- 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.
- And, hey, we're going to make pretty pictures, too.

Lab: Getting Started with Linux

- We'll break about midway through today's class to get you set up working with our Linux computers.
- This "lab prep" is somewhat 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. I understand that sometimes you have to miss. 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 quizzes
- Every Friday, plus a few pop quizzes
- The exams
    - Three take-home exams during the semester. Plan to spend four hours on each one.
    - An *optional* final to make up for a bad exam grade.
    - Take all three exams anyway.
- The labs
    - Available online.
    - I expect to require more formal writeups of about a lab per week.
- The homework
    - Weekly, due on Wednesdays
- Academic honesty
    - Core to the academic process.
    - Che college's basic policy: Cite carefully.
    - My basic policy: Don't cheat. (Also: It's usually okay to work with others, provided you cite them.)
    - Read my handout on academic honesty carefully.

 This material is based upon work partially supported by the National Science Foundation under Grant No. CCLI-0633090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.