University of British Columbia  
CPSC 111, Intro to Computation  
Alan J. Hu  

Course Information  
How Computing Happens  
Lecture 0  

Rule #1: Ask questions!  

Some slides borrowed from Kurt Eiselt and Steve Wolfman. Learning goal material from Beth Simon.  

Who I Am  
Alan J. Hu (You can call me Alan or Prof. Hu.)  
ajh@cs.ubc.ca  
ICICS 325  
office hours:  
I will hang out after class (both sections)  
if people have questions (MWF around 4pm)  
(TTh around 5pm).  
Feel free to stop by my office at other times,  
but I may be busy or not there.  

Other Introductions...  
Benjamin Yu, UBC  
Carl Weiman Science Education Fellow  
Beth Simon, UC San Diego  
Paul Denny, U of Auckland  

What This Course Is About  
Calendar description: Basic programming constructs,  
data types, classes, interfaces, protocols and the design  
of programs as interacting software components.  
Prerequisites: Mathematics 12.  

Who this course is for...  
Although it is expected that you will have used a  
computer prior to taking the course and that you are  
familiar with basic keyboard and mouse operations, no  
prior programming experience is assumed. This course  
will teach you basic programming constructs that will  
allow you to unleash your creativity and develop your own  
applications software.
...but note this

If you have already received credit for CPSC124 and CPSC126 or for CPSC122 and CPSC128 then you cannot receive credit for CPSC111. If you have taken CPSC124 but not CPSC126 then you must take CPSC111 in order to advance through the computer science program. If you have taken CPSC122 but not CPSC128, please consult with a department advisor.

Administrative Stuff

CPSC 111 has two sections taught in parallel:
- Same labs, tutorials, homeworks, and exams.

You should attend the lecture section that works for you. (Or more than one if you wish!)

Labs and tutorials start on Monday; more about these soon…

(Do Lab #0 on your own this week.)

Administrative Stuff

Your textbook is Big Java (3rd Ed). (The 2nd edition is OK, too.)

You should get a copy. Seriously.

Read Chapter 1.

Administrative Stuff

You will need: a Campus-Wide Login (CWL), and an account with the CS department.

You can find instructions for getting all of these on WebCT. This course is on the new Vista server: http://www.vista.ubc.ca

A good starting point is the course website, which will also get you to WebCT:
http://www.ugrad.cs.ubc.ca/~cs111/

Administrative Stuff

If you feel that you already know the 111 material, you can challenge the class. You must register for the challenge exam by Friday at 9AM to do so.

Unfortunately, taking the challenge exam costs as much as one credit (somewhere around $140). Read more at the course website!

Thinking about Thinking:

What do I need to know from today’s lecture?

- Studying computing is a new experience for you
  - Always know what you are supposed to be learning otherwise you might miss it!
- Learning Goals
  - Each class will start with a slide of learning goals for the days lecture
    - Help you identify what you should be “getting out” of lecture for the day
- ASK
  - If you don’t understand those as we go through lecture!
Learning Goals
By the end of class today you will be able to...

- Find out more detailed information about the course on-line.
- Say the name of your textbook.
- Find the CS Learning Center.

Course Learning Goals
- Be able to write simple (up to about a thousand lines long) programs that correctly express your intentions, in the Java programming language.
- Be able to read and understand the basic functionality of programs written in Java or similar programming languages.
- Change the way you think about computers!

This is a first course in computer science...
...but what is computer science?

"Computer science is as much about computers as astronomy is about telescopes." — Edsger Dijkstra

This is a first course in computer science...
...but what is computer science?

"Computer science is how to harness the physical world to aid or exhibit ‘thinking’." — Alan J. Hu

This is the most profound revolution in human history since the advent of writing!
The Most Profound Revolution…?

- Moveable Type Printing? Just lets you make more copies of some writing, more easily
- Steam Engine? Internal Combustion Engine? Just more efficient ways to harness physical objects to make us stronger.
- Boats, Trains, Planes? Just ways to move stuff somewhat faster and easier.
- Telephone, Radio, TV? Just ways to transmit information more easily.

Profound Revolutions

- Rocks, Sticks, Wheels, Levers: The first time humanity managed to use physical artifacts to augment our physical capabilities.
- Writing: The first time humanity managed to use physical artifacts to augment and encapsulate our memory.
- Computer Science: The first time humanity managed to use physical artifacts to augment and encapsulate thinking.

Harnessing the physical world to help us think…

- How to get create things that have "computational" behavior
  - Not the focus of this course
  - Technology dependent: sticks, gears, relays, vacuum tubes, transistors, DNA,…
  - How to control that behavior to do interesting things.

- Really simple example: slide rules
  - Physical lengths add/subtract.
  - What's something that lets you multiply/divide via addition and subtraction?

Harnessing the physical world to help us think…

- Another example: a toilet valve
  - This is a physical device that makes a decision!

  (In the 19th century, when the industrial revolution was going strong, a few people actually imagined building something like a modern computer out of steam-powered valves…)

- Another example: a relay
  - This makes a similar "decision" as a toilet valve.

  The first modern computers were built in the 1940s with relays…
Harnessing the physical world to help us think…

- Current technology: the MOS transistor
- This makes a similar “decision” as a toilet valve or a relay…

- Pretty much every computer or computer-like device is currently relying on MOS transistors…

**The MOS Transistor**

- **Depletion Region**
  - With no charge carriers in depletion region, electricity can't flow from source to drain.

- **Inversion**
  - A high voltage gives gate a positive charge…
  - … which attracts electrons to other side…

**Inversion**

- … which makes the p-type material act like n-type…

**Polysilicon**

**Silicon Oxide**

**Substrate (p-type)**
**Inversion**

A high voltage gives gate a positive charge...

... which lets electricity flow from source to drain!

**Computer Organization**

- What do you do with a few billion transistors?

- Some ideas have withstood the test of time:
  - Binary representation of data
  - General-purpose, *programmable* machine:
    - Memory, accessed by numerical addresses
    - Processor, which executes a sequence of instructions

**Binary**

- “Binary” just means that you have only two symbols, normally written 0 and 1.

**Binary Representation of Data**

- “Binary” just means that you have only two symbols, normally written 0 and 1.
- You can use binary to represent data in any way that you want:
  - 0 might mean “sun”, 1 might mean “rain”
  - 0 might mean the number 0; 1 might mean 1
  - 0 might mean the number 1; 1 might mean 0!
- We can choose what things mean, but we must be consistent.

**Binary Representation of Data**

- What if you want to represent more than two things? E.g. {sun, rain, snow}
- Must use more than one “bit” to represent your data!
  - 00 = sun
  - 01 = rain
  - 10 = snow
  - (11 = undefined)

**Binary Representation of Data**

- There are standard, widely used encodings for many things (positive integers, signed integers, real numbers, different alphabets, Chinese characters, etc.)
- But in the computer, it's all just a bunch of bits!
- Does 00 represent “sun”? Or the number 0?
Why Binary?

- It's easier to build a device that has to distinguish between only two values (instead of more values).
- E.g., with the MOS transistor, if the voltage at the gate is high enough, it conducts. If it's not high enough, it doesn't.

Computer Organization

- What do you do with a few billion transistors?

- Some ideas have withstood the test of time:
  - Binary representation of data
  - General-purpose, programmable machine:
    - Memory, accessed by numerical addresses
    - Processor, which executes a sequence of instructions
    - ...

Core Hardware Components (Highly Simplified)

- Input Devices
- Output Devices
- Central Processing Unit
- Memory
- Mass Storage Devices

Memory

Memory consists of a series of locations, each having a unique address, that are used to store programs and data.

When data is stored in a memory location, the data that was previously stored there is overwritten and destroyed.

Each memory location stores one byte (or 8 bits) of data. Each bit is a 0 or a 1 (more later).

Data values are stored in memory locations – more than one location may be used if the data is large.

What does a memory do?

- It sits around and remembers stuff…

- It responds to two kinds of requests:
  - Read: "Please tell me what you have stored in the memory location at address x."
  - Write: "Please write this data … to the memory location at address x."

- Who makes those requests?
The Central Processing Unit
The CPU executes instructions in a continuous cycle known as the “fetch-decode-execute” cycle.
The CPU has dedicated storage locations known as registers. One such register is known as the program counter which stores the address in memory of the next instruction to be executed.

Controlling the Computational Behavior
Because of the fetch-decode-execute cycle, we control the computer to make it do what we want by giving it a sequence of little steps (the instructions) for it to do.

Computer Programming

(Old Slide) What’s a Computer?
How is a computer different from a video game console? Or a DVD player? Or a telephone? Or a bank machine?
The computer is general. It can be all of the other devices.

Making the computer do what we want is called programming the computer.

Computer Programming

- You can make the computer do anything that it’s capable of. The only limits are space, time, I/O devices, and your skill and creativity
- It takes work.
  - The biggest program you’ll write in 111 will be a few hundred lines long.
  - Windows Vista is 50 million lines long.
- You have to write in a language the computer understands.

Questions?