University of British Columbia
CPSC 111, Intro to Computation
Wolfgang Heidrich

Introduction

Lecture 1

Borrowing from slides by Alan Hu, Kurt Eiselt, Paul Carter, and Tamara Munzner
Rule #1: Ask questions!
Who I Am

- Wolfgang Heidrich
  - you can call me Wolfgang or Prof. Heidrich
- Office: ICICS/CS x649
- Office hours: Friday, 2:00-3:00pm
  - (before class)
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...

- Expectations:
  - you have used a computer prior to taking the course and you are familiar with basic keyboard and mouse operations
  - no prior programming experience required

- Course goal:
  - you learn basic programming constructs that will allow you to unleash your creativity and develop your own applications software
...but note this

- You cannot get credit for both this course and CPSC 110!
  - Same material, but different programming languages
  - Since both courses are full, the department will only allow you to register for ONE of these
CPSC 111 has 4 sections taught in parallel:
- 3 “regular”, 1 BCS
- Same labs, tutorials, homeworks, and exams.
- Different times, professors, lectures, styles.
- You should attend the lecture section that works for you.

Labs and tutorials start on Monday
- However, lab 0 has to be done this week, on your own time
Administrative Stuff

- You will need: a Campus-Wide Login (CWL), an account with the CS department
- You can find instructions for getting all of these on WebCT or the course web page

- http://www.vista.ubc.ca
- http://www.ugrad.cs.ubc.ca/~cs111
Administrative Stuff

- If you feel that you already know the 111 material, you can challenge the class
  - Must register by Friday!
  - Details:

- The challenge exam costs the same as one credit (around $150)
Administrative Stuff

- Lab 1 (next week)
  - Regular lab problem (online, see WebCT)
  - 20 minute programming test
    - Simple programming test
    - Allows us to measure your baseline knowledge of computing – compare with results of similar test at the end of the term

- Preparation: read the handout material before your lab next week!
Administrative Stuff

- Textbook: Big Java, by Horstman
- Either edition 2 or 3 is fine
- Weekly reading assignments will start next week
This is a first course in computer science...

...but what is computer science?
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 the study of what computers do, not of what they are.”

Kurt Eiselt
This is a first course in computer science...

...but what is computer science?

“Computer science is how to harness the physical world to help us think.”

Alan Hu
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.
Computer Design

- Because it’s hard to figure out how to make things do computation, all digital computers for over 50 years have had:
  - Same basic organization
  - Binary representation of data
  - Numerically addressed memory
  - Fetch-decode-execute operation cycle
- Fascinating, but we’ll only glimpse...
Introduction to Computer Hardware

- Objectives:
  - to identify and explain the purpose of core hardware components
  - to understand the way data is represented in memory

Understanding the hardware that runs our programs can help us understand the programs' behavior, especially when they misbehave.
Core Hardware Components

- Input Devices
- Output Devices
- Central Processing Unit
- Memory
- Mass Storage Devices
Binary Data Representation

- All programs and data on a computer are represented using only symbols 0 and 1.
- This simple **binary** system is encoded in all of our digital hardware devices:
  - Magnetic disks: magnetic material can be polarized to one of two extremes (north or south) to represent a 0 or a 1.
  - Memory: each byte consists of 8 bits; each bit is a kind of electronic switch that is either off or on representing a 0 or a 1.
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.
## Units of Memory Storage

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
<th># of bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte (8 bits)</td>
<td></td>
<td>$2^0 = 1$</td>
</tr>
<tr>
<td>kilobyte</td>
<td>KB</td>
<td>$2^{10} = 1024$</td>
</tr>
<tr>
<td>megabyte</td>
<td>MB</td>
<td>$2^{20} = 1024^2$</td>
</tr>
<tr>
<td>gigabyte</td>
<td>GB</td>
<td>$2^{30} = 1024^3$</td>
</tr>
<tr>
<td>terabyte</td>
<td>TB</td>
<td>$2^{40} = 1024^4$</td>
</tr>
</tbody>
</table>
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.
Controlling the Computational Behavior

- A procedure is a collection of instructions in some meaningful order that results in useful behavior on behalf of the device that executes the instructions.

- When the instructions are written in a symbolic language that can be executed by a computer, the procedure is called a computer program.

- A process is what happens when a computer follows a program - it’s a procedure in execution.
Thinking in terms of process is crucial

- Formulas aren’t sufficient for describing how our world works. For example,
  - Economic systems are processes
  - Political systems are processes
  - How HIV invades cells is a process
  - How pharmaceuticals will interfere with HIV will also be a process

- Being able to think about complex systems in terms of procedures and processes will be of value to you even if you never write another program after 111.
Procedures and algorithms

- Computer people often use the words “procedure” and “algorithm” interchangeably... we will too.

- An algorithm is (Berlinski, The Advent of the Algorithm)
  - a finite procedure
  - written in a fixed symbolic vocabulary
  - governed by precise instructions
  - moving in discrete steps, 1, 2, 3, ...
  - whose execution requires no insight, cleverness, intuition, intelligence, or perspicuity
  - and that sooner or later comes to an end
Why computers sometimes drive us nuts...

- Computer people often use the words “procedure” and “algorithm” interchangeably... we will too.

- An algorithm is (Berlinski, The Advent of the Algorithm)
  - a finite procedure
  - written in a fixed symbolic vocabulary
  - governed by precise instructions
  - moving in discrete steps, 1, 2, 3, ...
  - whose execution requires no insight, cleverness, intuition, intelligence, or perspicuity
  - and that sooner or later comes to an end
How to avoid frustration

- Practice, Practice, Practice
- It takes a lot of practice to learn to be precise enough to make a computer do what you want
- It takes a lot of practice to keep from assuming that the computer is smarter than it really is
- It takes a lot of practice to get good at this stuff
Don’t wait until the last minute to get help
Bad things happen while learning a new skill. Start homework early; give yourself time for mistakes.

Tip #2

Hey, can I still pass if I can get enough partial credit?
Tip #3

Don’t be too ambitious with your course load. You can’t slack off in this class, even for a few days.
Coming Up

- Next lecture: taught by Cristina Conati (I will be at a conference)

- Topics:
  - Binary and hexadecimal number systems
  - Variables and assignment