Introduction

Introduction

- Administration
- Computers
- Software
- Algorithms
- Programming Languages

Course Website

- http://www.cs.sfu.ca/CC/130/johnwill/
- This course does not use Canvas

Success in this Course

- Come to class
- Read ahead
- Attend the labs
  - You get practice and can also ask for help
  - You get marks
- Complete the assignments
  - You get marks for them, and
  - You learn to program by programming
- Ask questions

Topics

- Programming in C++
- Problem solving, fundamental algorithms
- Elementary data structures
- Representation of Data
- Software design
- Algorithms

Assessment

Assessment

- Assignments – 25%
- Labs – 5%
- Midterm exam in class – 20%
- Final exam – 50%

Labs

- Labs are every week
  - Except for the first week
- Labs are usually assessed
  - This will be shown on the lab page of the site
  - Assessed labs are worth 0.5% of your final grade
- Labs are not exams – we want you to learn how to program – so ask for help!
  - And feel free to work with a friend

Assignments

- There will be five programming assignments
  - Each worth 5% of your final grade
- They must be submitted on the Computing Science submission server
  - Coursys
  - They must be completed on your own unless specified otherwise by the instructor (me)

Academic Honesty

- Read the [policy]
- Section 4.1.2 (e) forbids
  - Cheating in assignments, projects, examinations or other forms of evaluation by:
    1. using, or attempting to use, another student's answers;
    2. providing answers to other students;
    3. failing to take reasonable measures to protect answers from use by other students; or
    4. in the case of students who study together, submitting identical or virtually identical assignments for evaluation unless permitted by the course Instructor or supervisor.

Reasons Not To Cheat

carrie

- It's cheating and therefore immoral …
- Because we may catch you and then
  - You get zero on your assignment
  - You will get a letter on your file
    - And if you keep doing it
    - You get chucked out of SFU
- You won't learn how to program
  - Because you won't learn how to program you are far more likely to fail the final exam

Computers

And Computing Science

What's a Computer?

- Computers come in many shapes and sizes
  - Desktops
  - Laptops
  - Phones
  - Specialized systems
    - Braking systems
    - Toasters
    - Autonomous vehicles
- It's probably more useful to look at some characteristics of (modern) computers

Characteristics of a Computer

- Can do basic arithmetic (+, -, *, /)
  - Can perform these arithmetical operations *very* fast
- Represents data in binary
- Has a large main memory that can efficiently store and retrieve data
- Can accept input and produce output
- Can be programmed
  - Stores programs in main memory
    - One of the characteristics of the Von Neumann architecture

How Smart Are Computers?

- This course doesn't encompass philosophy
  - Do computers think, for example?
    - Which leads to, what is thought?
- Computers are very good at doing things that we find difficult to do
  - At least with any reasonable speed
- But does that mean that computers are generally "smarter" than people?

Computers

- Alienware Area 51
  - Uses Intel Core i7
  - ≈ 300,000 MIPS
- Lots of memory!
  - 32GB of RAM
  - 4 TBs of storage
- And it looks cool

People

- Human brain
  - Processing power estimated at 100 million MIPS
  - Memory estimated at 100,000 GB

What is Computer Science?

- It is the *study of algorithms and data structures*, including their
  - formal and mathematical properties
  - hardware realizations
  - linguistic realizations
  - applications

What is Computer Science?

- It is the *study of algorithms and data structures*, including their
  - formal and mathematical properties
    - What can be computed?
    - What is the most efficient way to solve a particular problem?
  - hardware realizations
  - linguistic realizations
  - applications

What is Computer Science?

- It is the *study of algorithms and data structures*, including their
  - formal and mathematical properties
  - hardware realizations
    - What's the structure of a CPU?
    - How is computer memory implemented?
    - What is the most cost-effective kind of computer hardware?
  - linguistic realizations
  - applications

What is Computer Science?

- It is the *study of algorithms and data structures*, including their
  - formal and mathematical properties
  - hardware realizations
  - linguistic realizations
    - What's the clearest/shortest way to describe computations?
    - What's the best way to organize large programs?
  - applications

What is Computer Science?

- It is the *study of algorithms and data structures*, including their
  - formal and mathematical properties
  - hardware realizations
  - linguistic realizations
  - applications
    - Graphics, artificial intelligence, databases, networking, software engineering, etc.

Software

Where Can We Find Computers?

- Telecommunications
- Medicine
- Information and Research
- Commerce
- Entertainment
- Finance
- Transportation
- …

Hardware and Software

- Hardware refers to computer equipment
  - Central Processing Unit (CPU)
  - Hard disk
  - Input devices (mouse, keyboard)
  - Output devices (printer, monitor)
- Software refers to the programs that give computers their behaviour

Software

- What is software?
  - A set of instructions for a computer
  - Programming therefore is telling the computer what to do
- Why is programming (considered) hard?
  - Because we want to solve hard problems
    - Usually things we can’t easily do by hand
  - And because computers are fundamentally stupid

Writing Software

- We write software to tell computers how to solve a problem
  - We've all given instructions before
    - Directions to a house
    - Using the microwave
- But, remember, computers are *<u>stupid</u>*
  - They can't deal with ambiguity
  - Instructions must be precisely defined in perfect grammar

So What?

- Though programs are written in an English-like language they are very formal
  - They must be written using correct syntax
  - They must be precise and unambiguous
- A program is a sequence of instructions that must be followed step by step
  - Implementation of *algorithms* that can be processed by a computer
  - Each instruction must be correctly specified for the program to function as desired

Algorithms

- A set of instructions for solving a problem
- Algorithms can be expressed in many different languages, for example:
  - English
  - A programming language (C++ for example), or
  - Pseudocode

Euclid's Algorithm

Input

positive integers *a* and *b*

Output

the greatest common divisor (GCD) of *a* and *b*

Algorithm

```
Repeat until $a$ and $b$ are the same value:
    if $a$ is greater than $b$:
        set $a$ to $a - b$
    else:
        set $b$ to $b - a$

Return $a$ as the answer
```

Try it when a = 91 and b = 65

Euclid Example

```
Repeat until $a$ and $b$ are the same value:
    if $a$ is greater than $b$:
        set $a$ to $a - b$
    else:
        set $b$ to $b - a$

Return $a$ as the answer
```

a	b
~~91~~	~~65~~
~~26~~	~~39~~
13	13

Result

Properties of an Algorithm

- Every step is unambiguous
- Input and output are clearly defined
- It must be executable in a finite amount of time

Properties of an Algorithm

- Every step is unambiguous
  - Because it is going to be run by a computer that is so stupid it makes Homer Simpson look like Albert Einstein
- Input and output are clearly defined
- It must be executable in a finite amount of time

Properties of an Algorithm

- Every step is unambiguous
- Input and output are clearly defined
  - Why? You know, can you, like, get me the thing over there and like make that other thing from it that we were, like, talking about the other day, you remember, the day when, like, that thing happened that was so cool, and you said "hey wouldn't be, like, neat if we had a thing like that".
- It must be executable in a finite amount of time

Properties of an Algorithm

- Every step must unambiguous
- Input and output are clearly defined
- It must be executable in a finite amount of time
  - If it doesn't run in a finite amount of time then waiting around for it to finish is going to be tricky
    - Even if you are immortal it's going to be very dull

Developing Programs

- Analysis
  - What is the problem?
- Design
  - What is the solution?
- Programming
  - Write the program
- Testing
  - Make sure the program works

Implementation

Programming Goals

- Correct
- Reliable
- Well designed
- Affordable
- Maintainable

Programming Languages

Types of Languages

- A program is written using a programming language
- There are different kinds of languages
  - Machine language
  - Assembly language
  - High level languages
    - C, C++, Lisp, Python, Java, Fortran, Perl, …

Machine Language

- Machine language is so called because it can be processed directly by a computer
- A program is a sequence of instructions
  - Each instruction code is represented by a number
  - Each number is represented in binary
    - i.e. 0s and 1s
- Machine languages are very hard for humans to write and understand

Assembly Language

- Assembly languages are human readable versions of machine code
  - Numeric codes are replaced by simple strings
- It is simple to convert an assembly language program into machine code

High Level Languages

- Assembly language is made up of very low level instructions
  - Simply adding two numbers may require four or five separate operations!
- High level languages are much easier to write and understand
  - Although their compilers are more complex
  - C++ is a high-level language

Formal Languages

- C++ is a high level programming language
  - It can be compiled into machine code
  - And executed on a computer
- Programming languages are formal and lack the richness of human languages
  - If a program is *nearly*, but not quite syntactically correct then it will not compile
  - The compiler will *not* "figure it out"

Brief History of C

- Create in 1972 by Dennis Ritchie of Bell Labs
  - When he and Ken Thompson were designing Unix
    - Developed from Thompson's B language
- Developed to be used as a tool for programmers
  - Working on low level system programs

Brief History of C++

- Started in 1979 when Bjarne Stroustrup wanted to extend C to use classes
  - First called *C with Classes*
    - In 1983 it was named C++
    - *C++ Programming Language* published in 1985
- In 2011 the C++11 standard was released with substantial changes to the language

Why C++?

- Efficient
  - Compact and runs quickly
  - In some ways similar to assembly language
- Portable
  - Programs run on one system can be run with little modifications on other systems
- Flexible
  - Allows programmers a lot of control
- Widely used

But …

- With flexibility and freedom come the possibility of more mistakes
- While compact it can be harder to learn with than more recent languages
- C++ has a variety of ways of achieving the same ends
  - Some are usually better, safer, or more efficient than others
  - It is a hard language to master