EN47/COMP9
Exploring Computer Science

Lecture 1

Introduction

Welcome!

Course mechanics

Number:	EN47-CS2 or COMP9-01, Fall 2009
Lectures:	Tue 3:00-4:15pm (Halligan 106)
Labs:	Thu 3:00-4:15pm (Halligan 116)

Instructor:	Sara Su (sarasu@cs.tufts.edu)
TA:	Mona Yousofshahi (myousofshahi@gmail.com)

Course website

http://www.cs.tufts.edu/comp/9

Bookmark it! We'll post announcements and handouts here.

Goals of this course

To gain an introduction to the field of computer science

To develop problem solving skills and computational thinking

To explore problem solving and programming techniques using the C++ language

Topics for today

Computer science

Computers

Problems and problem solving

Programming

What is computer science *not*?

Not just about building computers or writing code!

- These are the tools we use for computation

Learning to program is a good way to learn and practice the process of identifying problems and designing and implementing solutions.

Computational problem solving

Identify the problem

Design an elegant solution

Implement (automate) that solution

Discover and use the guidelines of how to compute, organize, and process information

Computation: need for consistency and repeatability

Computer Science

Study and applications
of the notion of **Algorithm**

Algorithm

"A precise step-by-step plan for a computational procedure that begins with an input value and yields an output value in a finite number of steps." (source: Wikipedia)

Areas of Computer Science

Here at Tufts, research areas include:

machine learning theory of computation

computer vision computer graphics

programming languages computational biology

algorithms computer architecture

robotics human-computer interaction

operating systems computational geometry

Theoretical concerns, as wells as systems and applications

Computer

A general purpose machine
for manipulating data

Computers are deterministic

Execute a set sequence of commands

Guided by current state and inputs

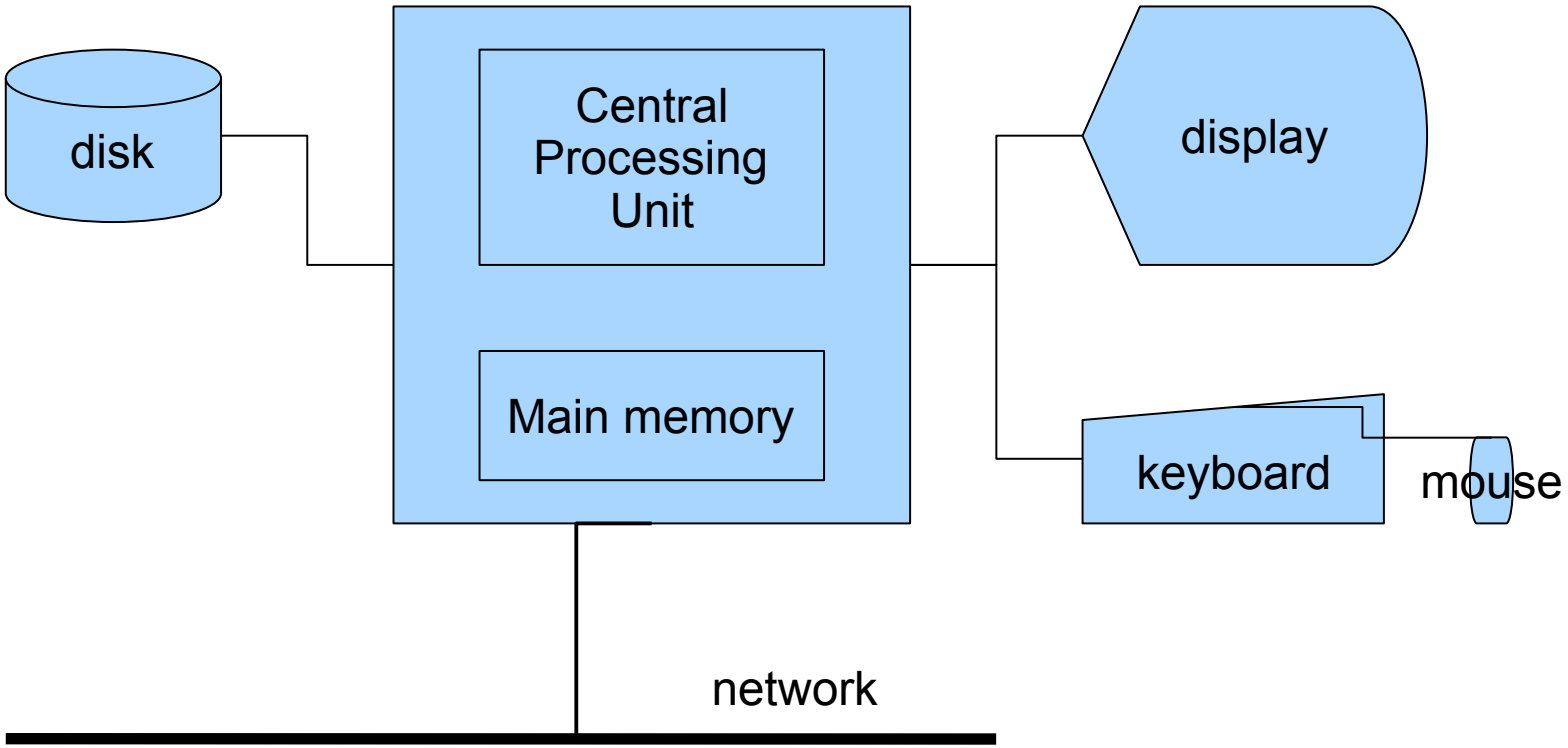
Always generate the same answer, given the same inputs

Computers are everywhere

How many computers have you used today?

How many are in this room?

A modern computer



Input devices

Send information to a computer

- Keyboard, mouse, game controller
- Scanner, camera
- Disk drives
- Network
- Punch cards

Output devices

Get information from a computer

- Computer monitor, printer, speakers
- Haptics devices (e.g force feedback game controllers)
- Hard drive, Writable CD drives
  - zip disk drives, floppy drives, tape drives, etc.
- Network

Central Processing Unit (CPU)

Two components:

- **Control Unit**
  - Retrieves and decodes program instructions
  - Coordinates activities of all other parts of computer
- **Arithmetic & Logic Unit**
  - Hardware optimized for high-speed numeric calculation
  - Hardware designed for true/false, yes/no decisions

Main memory

Volatile - erased when program ends or computer turned off

Also called Random Access Memory (**RAM**)

Organized as follows:

- bit: smallest piece of memory
  - Has values 0 (off, false) or 1 (on, true)
- byte: 8 consecutive bits - bytes have addresses
- words: size depends on architecture (e.g. 32 bits or 64 bits)
- subblocks
- blocks

Secondary storage

Non-volatile - data retained when program is not running or computer is turned off

Comes in a variety of media:

- magnetic: hard drive
  - Zip disk, floppy disk, tape
- optical: CD-ROM, DVD-ROM, Blue Ray-ROM (?)

Computers are great

Qualities:

- Precision
- Speed
- Size

Better than humans at:

- Sorting
- Searching, Retrieving
- Processing large amounts of data

Not as good as humans for other things...

Trends

Computers are getting faster and smaller fast!

- “Moore's Law”: number of transistors per chip doubles every 18 months, true over almost the past 35 years...

Computers beyond traditional electronics

- DNA computing
- Quantum computing
- Etc.

Computers: Summary

The modern computer is:

- Incredibly small, fast, and efficient
- **Deterministic**: always responds the same way given the same commands
- Capable of a variety of input and output
- Connected to the world
- *Nothing until a programmer tells it what to be!*

Problems

A wide variety of problems can be solved with computers:

- Mathematical calculations, simulations, graphics, data analysis, robotics control, trend prediction, scheduling, word processing, presentations

These problems are rarely expressed as sequences of bits.

What is programming?

Programmers stand between the world of problems and the world of the computer.

Take a problem

- Analyze the problem
- Envision a way to solve the problem

Take a computer

- Encode the solution
- Test & “debug”
- Maintain the solution

Programs

A program is a set of instructions to a computer

Computers are general-purpose machines

- Just about useless without a program

Programs turn computers into special-purpose devices capable of solving specific problems!

Programming skills

A mix of high-level creativity and low-level details

Programmers must learn to

- Be creative problem solvers
- Be meticulous artisans
- Think about computers at many levels
- Be patient and persistent!

Machine vs. high level languages

Computer hardware carries out instructions written in a **machine language**

- 1's and 0's – hard to understand or write!

A **high level language** is a notation that humans can understand and use more easily

A **compiler** translates a high level language to a machine language that can be executed

Evolution of languages

Languages change as needs and technologies change

- Machine Language - 1940's
- Fortran, Lisp - 1950's
- Cobol, Algol, APL, PL/I - 1960's
- Basic, Pascal, C - 1970's
- Smalltalk, C++, Modula, Ada, Prolog -1980's
- Java, Php, Python, Ruby - 1990's

Computer Science is important

Computers are everywhere, gaining ground, getting smaller and faster every year

Many of the problems that need solving today can be solved (or assisted) by computers

- Especially problems that are difficult for humans

The skills of programming enable us to create new solutions and better use existing ones

Course roadmap

Translate English into an **algorithm** and then into a **program**

- Which algorithm is more efficient?
- Will all programs eventually terminate?
- Cool programming assignments – simple to complex

Key concepts:

- The tools of programming
- The rules of programming
- The way of the programmer

Tools of programming

Techniques, ideas, terminology, and constructs that make programming possible and effective

- Problem analysis
- Divide-and-conquer
- “Conditionals”, “iteration”, “recursion”, etc.
- Test-case design

Rules of programming

We will use the C++ programming language

- Widely used and industrial strength

The constraints and syntax of C++ and computers in general

- Expressing programming constructs
- Understanding and debugging errors
- Familiarity with “artificial” languages

Way of the programmer

Style and patterns of thought that make programs successful and comprehensible

- Not “necessary” parts of the program
- Not required by the language
- Key to becoming a truly proficient programmer
- Key to creating elegant, beautiful code

EN47 / COMP 9: Summary

Prerequisites: No prior programming experience expected!

Fundamental concepts and algorithms

Programming assignments (in class) using C++

Collaborative, class-wide final project

Reading

Required text:

Allen B. Downey, *How to Think Like a Computer Scientist: C++ Version*, Green Tea Press, 1999

available 9/10 at Gnomon Copy

Evaluation

Grading (approximately):

- 65% Assignments
- 25% Final project
- 10% Class participation

Expectations:

- Ask questions
- Help one another, but do not share code
- Get ideas from books/online, but always cite sources
- Challenge yourself

Accommodations

To request an accommodation for a documented disability, you must register with the Disability Services Office at the beginning of the semester.

Call (617) 627-2000 to arrange an appointment with Sandra Baer, Program Director of Disability Services.

This Course: Summary

Some tools

Some rules

And the way is up to you...