CMPSC 16
Problem Solving with Computers I
Spring 2014

Instructor: Lucas Bang

Lecture 1: Introduction
What is a computer? Some History

Babbage’s difference engine, 1853
What is a computer? Some History

Colossus, 1943

ENIAC (Electronic Numerical Integrator and Computer), 1945
What is a computer? Some History

IBM mainframes of 1960s and 1970s
What is a computer? Some History

IBM PC and Apple Macintosh from 1980s
What is a computer? Nowadays

- tablets
- smartphones
- laptops
- connected via internet
- datacenters
- desktops
What is a computer? Nowadays

Cars also have computers in them

Planes also have computers in them

Possible future: Almost everything will have a computer in it connected to the network
What is a computer?

We have seen many examples of computers...

... but, what is a computer?
What is a computer?

- A computer is a **programmable** device

- A **program** is a set of instructions that tell computer how to behave
  - So, a computer is something whose behavior is modifiable by providing a program

- After giving a computer a program, we can ask the computer to follow the instructions of the program
  - This is called **executing** or **running** the program

- When we run the program on a computer, the computer behaves exactly as instructed by the program
Why call it computer?

• A **computation** is producing an output from a set of given inputs based on a given program

• So a computer is the thing that does computation

• Also “programmable device” does not sound as good as “computer”

• What are some more examples of computers?
Why call it computer?

- The word “computer” originally referred to a person who “computes”.
- “Computer” was a job title.
Organization of a Computer

• We said that a program is a set of instructions, at its core the computer has a **processor** that executes instructions
  – Processor is the thing we are instructing what to do by writing a program

• During computation a computer may have some intermediate results that it needs to store
  – It needs a **memory** to remember these intermediate results
Organization of a Computer

• A computer needs to interact with the outside world

• A computer needs way to **receive inputs**
  – keyboard, mouse, touch-screen, microphone, camera, etc.

• A computer needs ways to **report outputs**
  – screen, printer, speakers, etc.
Internal Organization of a Computer
Internal Organization of a Computer

• Computer’s memory can have different types:
  – Memory that can be accessed fast which can only be read and cannot be written: Read Only Memory (ROM)
  – Memory that can be accessed fast but is deleted when the power is turned off: Random Access Memory (RAM)
  – Memory that is slower to access but can store the data even if the power is turned off: hard disk, extern drives, thumb drives

• Computer processor can be divided to specialized units
  – Arithmetic Logic Unit (ALU) that only specializes in arithmetic calculations

• Computer can have co-processors that do specializes computations
  – Graphics coprocessor
Hardware vs. Software

- The physical parts of the computer is called the **hardware**
- The programs that run on the computer are called the **software**
- Programs, code, software have similar meanings
- The person who writes programs or software is a **programmer** or a **software developer**
What is this class about?

• This class is about learning how to program

• Programming is not easy

• Here is what I read in the news this morning:

  "Technical problems pose roadblock to those looking to beat ObamaCare deadline"

  *Healthcare.gov was down for several hours Monday as last-minute applicants flocked to the site on the last day for signups. A software bug knocked the website offline around 3:20 a.m. Monday. When trying to access the log-in on Healthcare.gov, users were met with an error page that read, "The system isn't available at the moment. We're currently performing maintenance. Please try again later."

  Monday is the final day to sign up for health insurance for 2014."

• A **software bug** is an error in a program
How do we program?

- Computer scientists developed programming languages to help software developers in writing programs.

- **Machine languages**: This is the language of the microprocessor, it tells microprocessor what instruction to execute, written as 0s and 1s.

- **Assembly languages**: This is very close the machine language, but instead of 0s and 1s it is written typically using ASCII characters to be more readable for humans. An assembler translates it to the machine language.

- **High level programming languages**: These are programming languages that are developed to make it easier for humans to write programs. We are going to learn one of these languages in this class.

- High level programming languages contain **libraries** that are programs that handle common tasks (such as printing to screen) which make writing programs easier.
We will learn a high level language

- In this class we are going to learn a high level programming language called C.
- C language programs are translated to the machine code using a program called a compiler, so that it can run on the computer.
- The compiled program then is linked with the libraries, and then it can be executed on a computer.
Software Interface to the Computer
Developing software takes a lot of effort.

One can identify several phases in software development:

- **Requirements analysis**: Figuring out precisely the task that the software that we plan to develop is supposed to do.
- **Software design**: Coming up with a high level solution to how to do that task in a program.
- **Software implementation**: Translating the high level design into a program written in a programming language.
- **Testing and debugging**: Checking the program to identify and eliminate errors.
- **Maintenance**: Improvements and modifications to the program after it is released.

All these phases together are called the **software life-cycle** or the **software process**.
The process suggested in the Textbook

The textbook suggests the following steps for programming tasks:

• Problem statement
  – starts with clearly defining the problem to solve

• Input/output description
  – treats solution like a “black box” – focus on what goes in and what comes out

• Work the problem by hand on an example
  – two purposes: (a) verify you know exactly what to do; (b) have known results to check solution later

• Algorithm development
  – includes designing the solution and implementing it

• Testing
  – making sure solution works correctly
About this course

• Instructor: Lucas Bang
• Teaching Assistants:
  - Omer Acerol
  - Arvind Chitra Rajasekaran

• Class Times: Monday/Wednesday 2:00-3:15
• Location: PHELPS 3515

• Lab Times:
  - Tuesday 12:30 - 1:45 Location: PHELP 3525
  - Tuesday 2:00 - 3:15 Location: PHELP 3525
  - Tuesday 4:00 - 4:45 Location: PHELP 3525
About this course

• **Important assumption:**

• Students who plan to take this course should be able to program in some high-level language, so already familiar with fundamental concepts:
  - Data types, data storage (memory), variables, constants, assignment, arithmetic (including integer arithmetic)
  - Logical operations, selection and iteration structures
  - Modular programming (functions/methods)
  - Array basics
About this course

Covers all of textbook (and some stuff not in text)

• Introduction to problem solving / C (Chapter 1)
• C programs, data types, operations (2)
• Control structures, data files (3)
• Functions and testing (4)
• Arrays (5)
• Pointers (6)
• Structs (7.1-7.5)
• Linked lists (7.6)
• Introduction to C++ (8)
About this course

Course work:
• One homework + lab each week
• Three programming projects
• Two midterms and a final

• All assignments, and some announcements, will be on the class webpage, it is your responsibility to monitor it:

   http://www.cs.ucsb.edu/~bang/cs16/

• Class discussion forum:

   https://piazza.com/ucsb/summer2014/cs16/home
Things to do this week

• Do first homework (Hw0) before lab

http://www.cs.ucsb.edu/~bang/cs16/assignments.html

- Very important: Get your computer science account.

https://accounts.engr.ucsb.edu/create/

• Go to your lab section

• Read text chapters 1 and 2