Foundations of Computer Science
Lecture #1: Introduction

Anil Madhavapeddy & Amanda Prorok

2019-2020

Getting Started

- Course Home:
  https://www.cl.cam.ac.uk/teaching/1920/FoundsCS/
- Interactive online notebook:
  https://hub.cl.cam.ac.uk/
- This notebook corresponds to the printed notes that you should all have.
  If you cannot login, email us immediately.
- At the end of this lecture, will also explain the practicals:
  https://www.cl.cam.ac.uk/teaching/1920/OCaml/

- **Computers:** a child can use them;
  but nobody can fully understand them!

- We can master complexity through *levels of abstraction*

- Focus on 2 or 3 levels at most!

- **Recurring issues:**

  - *what services* to provide at each level

  - *how to implement them* using lower-level services

  - *the interface* by which two levels should communicate

Example: Dates

- **Abstract level:** dates over a certain interval
- **Concrete level:** typically 6 characters: YYMMDD
  - (where each character is represented by 8 bits)
- Date crises caused by inadequate internal formats:
  - *Digital's PDP-10:* 12-bit dates (good for at most 11 years)
  - *Y2K crisis:* 48-bits could be good for lifetime of universe!
- Our choices of representations within a computer has long-ranging consequences.

Example: Floating Point Numbers

- Computers have *integers* (like 1066)
  and *floats* (like 1.066 × $10^{3)}$.
- A floating-point number is represented by two integers.
- The concept of a **data type** involves:
  - how a value is represented inside the computer
  - the suite of operations given to programmers
  - valid and invalid (or exceptional) results, such as “infinity”
- Computer arithmetic can yield incorrect answers
  due to **finite precision**!

Goals of Programming

Goals of Programming

- to **describe a computation** so that it can be done *mechanically*:
  - *expressions* compute *values*
  - *commands* cause *effects*

Goals of Programming

- to **describe a computation** so that it can be done *mechanically*:
  - *expressions* compute *values*
  - *commands* cause *effects*
- to do so **efficiently and correctly**, giving right answers *quickly*

Goals of Programming

- to **describe a computation** so that it can be done *mechanically*:
  - *expressions* compute *values*
  - *commands* cause *effects*
- to do so **efficiently and correctly**, giving right answers *quickly*
- to allow **easy modification** as our needs change
  - through an orderly *structure* based on *abstraction* principles
  - programmer should be able to predict effects of changes

Why Program in OCaml?

- It is **interactive**.
- It has a flexible notion of **data type**.
- It hides the underlying hardware: **no crashes**.
- Programs can easily be **understood mathematically**.
- It **distinguishes naming** from updating memory.
- It **manages storage** in memory for us.

The Practical Classes

https://www.cl.cam.ac.uk/teaching/1920/OCaml/