Programming

Imperative

Program = Sequence of commands/instructions
= a sequence of assignments [I/O]

```
x := 0;                    ← x is a variable
While (x ≤ 10) do
    x := x + 1;
    Print("boo")
```

If one wonders what x is, one has to know at what time.

Functional

5 + 3 = 8 — Symmetric (a = b iff b = a)
⇔ — Direction of operation
(5 + 3) × 4 → 8 × 4 → 32, We know that 32 is "the end". It's a canonical value, a known value.
Expression → Canonical Value
(simplify)

Example: Factorial
fac 0 = 1
fac n = n * fac(n-1)

Maths (logic)

1. Let x be the no. of apples: $x - 2 = 3 \Rightarrow x = 5$
   x is unknown
2. $S = 16t^2$
   Placeholder

Example - URE

Typically applied on payroll
One card per employee input (200 cpm)
Process info (100 rec/min, avg)
Print salary info (300 lpm)

```
loop:
    read card;
    process info;      (CDR → CPU → LPR)
    print
```

This is sequential CDR waits while processing+printing. How to speed up? How do we get them all to work at the same time?

If they all worked at the same pace this wouldn't be such an issue.
CDR → CPU → LPR, CDR → CPU → LPR, CDR → CPU → LPR, ...
1 x x 2 1 x 3 2 1

But obviously this isn't the case.

- We said the CDR waits, but does cards wait? See slides, lecture 2. Ex URE
  - Active / Passive
    Where does action come from?
  - CDR, LPR and CPU act. Each has private memory.
    They share info: how?

CDR puts contents in shared memory. By interrupts the CPU knows that contents has arrived.

CDR	CPU	LPR
loop	loop	loop
c := card	p = f(c)	paper := p

We need timings.

How to sort a bunch of kids

How to find the tallest kid:	1) Candidate 2) Tournament	1: 2: m, n: m if m ≥ n Guarded chemical equation
How to sort them:	1) Bubble-ish	1) $(i, h_i), (j, h_j) \rightarrow (i, h_j), (j, h_i)$ if $h_j > h_i$

Meta rules: $m, n \equiv n, m$

If an equation is enabled, it must eventually run and complete

Observations

Concurrency is simpler: Don't need explicit ordering
The real world is not sequential

Concurrency is harder: Many paths of computation (bank example)
Cannot debug since it's not deterministic, proofs are needed.

Time, concurrency, communication are issues

Terminology

A "process" is a story. A sequential component that may interact or communicate w/ other processes.
A concurent "program" is a BIG story. It's built out of component processes.
The components can potentially run in parallel, or may be interleaved on a single CPU.