Folk Chapter 1

Intro to the design and specification of file structures

Disks

- Disks are **very slow** to access *compared* with RAM
  - Nano- vs. milli- sec.
- RAM – volitile
- Disk – non-volitile
- Disks have *higher* storage capacity than RAM
  - giga- vs. tera-byte

File Structures

- Def: **Data** in files *and* **operations** for access
  - Read
  - Write
  - Modify
  - Order
- Details of *representation* and *implementation* determine efficiency
- What's *best* for one app. may be *worst* for another

File structure design course objectives

- Think creatively about file structure design problems
- Historical presentation of developments
- Design Goals:
  - Single disk access for desired data
  - If not, close to single
  - Aggregate data for multiple data w. 1 disk visit

File structure challenges

- Files – *static* vs. *dynamic*
- Ordering:
  - Sequential (usually sorted)
  - Indexed
  - Recursive graph structures (e.g. trees)
    - Binary: unbalanced vs. balanced (e.g. AVL)
    - N-ary: (e.g. B-tree)
  - Randomizing probabilistic structures (hashing)
  - Hybrids of the above (e.g. B^+tree)

File structure literacy

- **Evolution**: *Problem – Solution* cycle: the solution to one problem introduces new problems to solve
- Basic conceptual design tools:
- Aggregation of small things
- Splitting of large things

Object-Oriented Design

- **Class** definition: unified presentation coupling data types and the operations (functions or methods) on those data types
- Each file structure approach is represented by its own class
- Complications: Classes in practice are *progressive*, i.e., modifications or extensions of other related classes

Objects in C++

- `struct`-*like* syntax
- Semantic components:
  - Definitions
  - Constructors and destructors
  - encapsulation
  - Access privileges: private, public, friend, protected
  - Scope resolution (`::`)
  - Operator overloading
  - Inlining
  - Self-reference (`this`)