Bradon Ladd

4-1-19 (Day 1) // Sequential and Parallel Data Structures and Algorithms

**Lecture Notes**

-Slides n’ stuff

-Basic introduction to the class

**Reading Notes**

Chapter 1 – Software Engineering Principles, talks about the various design principles in software engineering. It’s broken down into: The Software Process, Program Design, Design Approaches, and Verification of Software Correctness.

*The Software Process*

Hacking away at programs until they work is not an efficient way to design programs, and should be unlearned early on. Software Engineering refers to the discipline concerned with all aspects of the development of high-quality software systems, and utilizes the software process, a specific set of inter-related software engineering techniques (this will be important to learn). *Quality software works, can be modified without excessive time and effort, is reusable, and is completed on time and within budget*.

Two design models: Waterfall and Spiral. Waterfall goes as such:

Analysis ->Requirements Identification ->Design ->Implementation ->Testing ->Delivery ->Maintenance

Spiral is less used and not required to learn.

Every programmer needs a toolbox consisting of hardware, software, and idea-ware (shared knowledge of algorithms and methods). This toolbox helps us achieve quality software, which achieves the goals that it works, it can be modified without excessive effort, it is reusable, and it is complete on time and within budget. Programs need to be *complete, correct, usable, and efficient*. When given an assignment that needs to meet these criteria, and the deadline is fast approaching, don’t’ panic. Instead, stop and think. Understand the problem. All of this is good pre-requisite thinking for program design.

*Program Design*

We use models to represent the inner workings of programs, such as abstractions, or models that only show what needs to be shown in order for the program to work. Whole models are broken down into modules, subunits of the program that perform a function, and can be added or removed as necessary (essentially a grouping of like-focused code), which can also utilize a form of abstraction called information hiding. Good system design results in the creation of generic modules that can be used in other systems.