**CSCI 3302 Lesson 1 – Introduction to Data Structures**

**Learning Objectives:**

* Understand the purpose and organization of the course.
* Define abstraction and information hiding.
* Describe the concept of an Abstract Data Type (ADT)

**Readings:**

* Review Chapter 1 of the text.
* Read Chapter 2, Section 1 of the text.

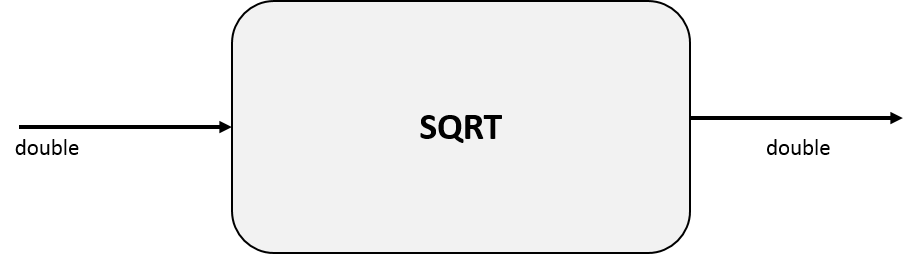
**Course Admin (30 minutes):**

* Slides – Lesson 1
* What is a data structure?
  + What this course is about and how it fits into total CS education.
  + "What is a data structure?"
  + Example: Library setting. How are the books organized? Alphabetically? Topically? By the color of the book jacket?
    - In a bookstore, fiction is arranged alphabetically by author, whereas non-fiction is typically arranged topically.
    - There is an intuition about how someone might want to find books. If they liked the last Stephen King novel, perhaps they are searching for more books by this author. If someone is looking a travel guide for their trip to Italy it would be helpful for all these types of books to be on the same shelf.
  + The analog in computer science is that when we want to access data within a program, we would like it to be organized efficiently based on how we intend to use it. Do we need to access all of our data sequentially? Do we need to be able to quickly look-up a specific piece of data (or perhaps we just want to know if a piece of data is there or not)?
* Review course syllabus:
  + Summarize course objectives
    - Briefly summarize the course objectives. They really boil down to 3 main points. We want students to be able to
      1. Design/implement data structures on their own,
      2. Use appropriate data structures to solve problems (esp. those that would come standard with a high-level programming language), and
      3. Become a better programmer overall.
  + Prerequisites
  + Office hours
  + Textbook and readings
  + Homework
    - Must compile/run for any credit
    - Individual effort/Paired Programming
  + Student expectations
* HW: Watch videos and load first assignment

**Lesson (45 minutes):**

* **Problem Solving and Software Engineering:**
  + **Problem solving** refers to the entire process of taking the statement of a problem and developing a computer program that solves that problem.
    - Typically, a **solution** consists of two components: algorithms and ways to store data.
    - When constructing a solution, you must organize your data collection so that you can operate on the data easily in the manner that the algorithm requires.
    - Most of what we will do in this course is about how to organize the data.
  + An **algorithm** is a step-by-step specification of a method to solve a problem within a finite amount of time.
    - One action that an algorithm often performs is to operate on a collection of data

* **Abstraction and Information Hiding:**
  + What is abstraction? We can think about how a car works at different levels of abstraction.
    - At the level of the operator, we insert the key and turn, use the steering wheel, foot pedals, and gear shift.
    - At the level of the mechanic, we understand how the various parts work together. The fuel injection system loads the cylinders with gasoline; the spark plugs ignite the fuel which turn the pistons; this turns the crankshaft which turn the wheels, etc.
    - At the level of the physicist, we understand the chemical process of adding a spark to fuel; we can calculate the force generated; we can measure the heat build-up and the friction of the piston in the cylinder.
  + When you go through the drive-through at McDonalds, you probably don’t care exactly how many cooks are on duty, or the arrangement of the patties, lettuce, and tomatoes around the griddle. What you care about is the input (the cost of the food; the time spent getting it) and the output (the quality of the food; the friendliness of the cashier).
  + **Procedural abstraction** separates the purpose of a method from its implementation.
    - The idea here is that what is done is important and how it’s done is not as important.
    - Think about the Math.sqrt() command. We can visualize it as a black box where we provide a number as input and we get a number as output.



We don’t need to know how it works in order to use it. We could, perhaps, write our own method using an implementation of the Newton-Raphson method. But even then, we may not have a good understanding of how the method is actually operating at the machine level (i.e., what are the machine instructions that were generated by the compiler). We may not understand how all of this is actually being carried out at the physical level with electrons and silicon gates. The ability to abstract away these lower details makes our job easier as a programmer.

* Going up in abstraction, instead of down, think about a large programming project with multiple programmers. Procedural abstraction is essential to team projects. After all, in a team situation, you will have to use methods written by others, frequently without knowledge of their algorithm.
  + **Data abstraction** focuses on what the operations do instead of on how you will implement them. In this form of abstraction, instead of just focusing on operations, we focus on data first and then the operations that manipulate the data.
* An **Abstract Data Type (ADT)** is a collection of data and a set of operations on the data.
  + Enables you to focus on what operations you will perform on the data instead of how you will perform them.
  + Ultimately, someone—perhaps you—will implement the ADT by using a data structure, which is a construct that you can define within a programming language to store a collection of data.
  + The principle of **information hiding** helps you to think about what details of a module should be visible from the outside, or **public**, view and what details should be kept **private**.
    - Information hiding limits the ways in which you need to deal with methods and data.
    - As a user of a module, you do not worry about the details of an implementation.
    - As an implementer of a module, you do not worry about its uses.

D2L Brightspace (10 minutes):

* Organization of course website on Brightspace