SIENA*college*
Computer Science

Computer Science 210
Data Structures
Siena College
Fall 2020

Topic Notes: Introduction and Overview

Welcome to Data Structures!

Why Take Data Structures?

You have programming experience, and for most if not all of you, at least some of it is in Java.

In this course, you will become a more sophisticated programmer and problem solver, as you learn about designing correct and efficient algorithms and data structures for use in your programs. Along the way, you will:

- hone your problem solving skills,
- gain experience in programming in general, Java in particular,
- learn how to implement algorithms and data structures in Java,
- learn how to evaluate and visualize data structures and algorithms,
- learn how to understand (and prove) some properties of data structures and algorithms,
- learn how to consider the relative merits of different structures and algorithms, and
- learn how to design large programs (in an object-oriented way) so that it is easy to modify them

By the end, you will be ready to study the more advanced programming techniques and constructs in CSIS 225, Advanced Programming, and more advanced algorithms in CSIS 385, Analysis of Algorithms. In addition to those two courses, this course is the prerequisite for many other mid and upper level courses in Computer Science.

We focus mostly on the relatively simple textual interface often used by advanced programmers as opposed to graphics. But the algorithms and data structures may be used in (and are often essential to) those graphical programs, and some of our programs might use a graphical user interface. Your additional programming experience will allow you to understand and make use of the extensive base of reusable code, Java and otherwise, that is available to today's programmers, even though we will use only a limited subset of those tools here.

Why is Programming So Challenging?

What you are asked to do in this class is very challenging. Introductory programming is challenging, but now that you've gotten through that, you are ready to move more quickly.

It takes extensive practice to become a highly skilled programmer. And the only way to get there is to do a lot of programming. So we will.

But what is it that makes programming such a challenge? Think about what you're being asked to do:

1. Understand the task to be accomplished at a high level (*i.e.*, "we want to find the longest word in this text").
2. Identify and understand the low-level specifics of the task (*i.e.*, "what do I do if there is a tie for the longest word?").
3. Formulate an algorithmic approach to accomplish the task for which you can develop a mental model, but more importantly, describe very precisely in English.
4. Determine the key variables and data structures needed by the algorithmic approach.
5. Write correct high-level language code to create and manage the variables and data structures and implement the algorithm.
6. Document and test your implementation.

In many cases, it's glossing over that third step that causes trouble in the actual design and implementation of the code. If you can't describe precisely what you are trying to do in your native language, be it English or some other spoken and written language, how can you possibly do so in the foreign language you are trying to learn? No one is a "native speaker" of Java, C, Python, or any other programming language. And that foreign language, whatever programming language you are using, is very picky and will not tolerate the kind of mistakes, say, that a native speaker of a spoken foreign language can tolerate when trying to understand someone who speaks their language very poorly.

We can think of a program as such:

Program = Algorithms + Data Structures

As a beginning programmer, you learn to think algorithmically. In fact, early programming languages had limited support for data structures.

Modern programming, and much of our focus for this class, is the *object-oriented* programming model.

- Code re-use, smaller modular code
- *Encapsulation* of variables (data) and code (methods)
- The choice of data structure determines what may be done to data and how efficiently

- Four basic functions of persistent data storage: the "CRUD" model
  - Create, Read, Update, Delete

Adminstrative Tasks

(See syllabus and course web site.)

Advice

Important: check your school email regularly. That will be my primary way of getting you important information. If you don't check it regularly, forward it somewhere that you do check regularly.

General important note: you will probably get confused at some point, but *don't remain confused*. If you do:

- come to my office hours or make an appointment
- e-mail any time
- see the tutors (they start next week)
- head off the problems before they get severe
- make me work – that's why you're here

I expect that I will see many of you in my office on a regular basis.

Coming to office hours will never be seen as a sign of weakness. On the contrary, asking for help outside of class will be seen as a sign of dedication to the course and a commitment to your success.