CS 300
Data Structures and Algorithms

Neeraj Jaggi
neeraj.jaggi@wichita.edu

Prerequisite Review (CS 211/EE 138)

- Data Types
  - int, char, float, double, long int, void
- Operators
  - ==, <=, &&, &, >>, +=, ++, &, *, ? :, ., ->, (int), sizeof
- Variables and constants
  - Scope
- Arrays
  - Fixed size, indexed access
- Statements
  - If, case-switch
  - Assignment

Prerequisite Review (contd.)

- Structures and unions
  - Heterogeneous fields
- Functions
  - Arguments, return value
  - C – pass by value
- Loops
  - while, for, do – while
  - break, continue
- Pointers
  - Address of object or function, arithmetic
  - Dynamic memory allocation

Lab Information

- Programming Assignment #1 (Due 02/03)
- Guidelines (Handout)
  - vi editor
  - programming style recommendations
- Lab account
  - Don't have username – contact instructor
  - Password
    - Don't use poor (weak) passwords
    - Forgot password – contact instructor
    - Default password (student initial + birth date)
      - Sample Username: rxrandom
      - Sample Birth Date: 1984-1-1
      - Sample Default Password: rxr19840101
  - Unix usage – Short Guide (collect from me during lab/office hours)
  - FAQ – http://www.cs.wichita.edu/~wallis/asksystem.html
  - Programming Assignment Submission – http://www.cs.wichita.edu/~wallis/handin/

Data Structures

- What
  - Method to store data *efficiently*
  - Primary design consideration in a computer program
  - Examples – Array, Structure in C
- Why
  - Accessibility
    - Time to access the data (find)
  - Maintainability
    - Ability to manipulate the data (add, delete, sort)
  - Reduce execution time and memory space requirement
- Where
  - Example – Dictionary, Database
  - Choice of data structure – Application specific
  - Tradeoff between access time and storage requirement

Algorithms

- What
  - Sequence of finite, well-defined instructions which achieve a specific task
  - A computer program is an algorithm
    - Should eventually halt !
- Why
  - Design of algorithm is important to ensure *efficiency*
  - Efficiency
    - Computational Complexity (execution time)
    - Storage requirement (memory space)
  - Choice of algorithm – Application specific
- Where
  - Example – Euclid's algorithm to find GCD of two numbers
  - Sorting, Searching, Find shortest path in a network

Course Overview

- Basics
  - Pseudo code, ADT (Chapter 1)
  - Recursion (Chapter 2)
- Well known Data Structures
  - Stacks (Chapter 3)
  - Queues (Chapter 4)
  - Linked List (Chapter 5)
  - Trees (Chapter 6)
  - Binary Search Trees (Chapter 7)
- Common Applications
  - Searching, Hash Tables (Chapter 13)
  - Graphs (Chapter 11) (time permitting)
  - Sorting (Chapter 12) (time permitting)