Data Structures

Topic #1

Welcome!

Today’s Agenda

- **Introduction**...what to expect!?!
- Talk about our **Goals** and **Objectives**
- **Textbook** is highly recommended
- **Lecture Notes** at clean copy
- Discuss what **Assignments** will be like
- This week is mostly review, so we will go rather rapidly through the material...and then slow down

Programming Paradigms

- ***Procedural Abstraction***
- ***Modular Abstraction***
- ***Data Abstraction***
- ***Object Oriented Programming***

Data Structures

- Our goal this term is to spend our time talking about different data structures, algorithms to solve problems, and how to measure the efficiency of the approaches taken
- This term is not about learning new C++ syntax!
- Instead, we will apply C++ and linked lists to new **abstract data types**

Data Structures vs. ADT?

- So, what is the difference between a data structure and an abstract data type?
- A data structure specifies **how we store the data** (like an array, linked list)
- An abstract data type **(ADT)** specifies how a new data type behaves: it includes the data and operations that the new data type requires; the data being stored in a data structure!

Data Structures vs. ADT?

- We will be building **<u>Abstract Data Types</u>** all term!
- Let’s walk thru some examples of ADTs
  - int
  - list
  - stack
  - queue

Data Structures vs. ADT?

- So, what is the difference between:
  - Data Abstraction
  - Abstract Data Type
  - Data Structure
  - Client vs. Client Program
  - User vs. Application

Using Classes to build ADTs

- We will use the C++ class construct to build abstract data types
- The data (represented by a data structure) are placed in the private section
- The operations (what the “client” or application can do) is in the public section

Using Classes to build ADTs

- The user & client should not be aware of what data structure is being used
- This means the client program should not be aware that there is a node or a next pointer for a linked list, or an index to an array -- if an array is used
- This allows an ADT to “plug and play” different data structures, to maximize efficiency w/o disrupting the client program

Using Classes to build ADTs

- Given this, what do you think we should do about:
  - data members?
  - error messages?
  - input of data?
  - output of data?
  - prompting?

Using Classes to build ADTs

- For each data member, ask yourself the question....could this be a local variable to a member function instead?
- If the value of the variable does not need to persist from operation to operation, it should <u>not</u> be a data member!

Using Classes to build ADTs

- The client program represents...
  - your “test bed” or the “application program”
  - represented by your main program
- Keep your data members restricted to just what is necessary
- Classes should not prompt
- For 163, classes should also not perform input...we will change that in CS 202

Using Classes to build ADTs

- The main program is the only place you should use statically allocated (do you remember what this term means?) arrays
- All arrays <u>must</u> be dynamically allocated in your class...why?
- Think about the when an ADT is written vs. an application. The ADT should be able to be used by many applications...

Using Classes to build ADTs

- Try to make your ADT’s as general as possible (without getting into templates).
- This means don’t tie the member functions to reading information from the keyboard
- Because...the ADT doesn’t know if there is anyone AT the keyboard!!!

Using Classes to build ADTs

- Use arguments instead and have the main() read from the keyboard (and prompt)
- This way, information can come from the keyboard or a file!!!
- And, of course, no global variables are allowed
- And, never prompt from a member function! Think of an int prompting!!!!!

Using Classes to build ADTs

- Think about Efficiency Too this term!
- Only traverse lists when absolutely necessary
- Use pass by reference to reduce the information from continually being copied -- when passing instances of a struct or a class
- And, remember to wear “different hats”