Review for Midterm

Chapter 1-9

CSc 212 Data Structures

Chapter 1

- Course Objectives
  - WHAT (Topics – ADTs, classes)
  - WHY (Importance – not only for credits)
  - WHERE (Goals – data structure experts)
  - HOW (Lectures, Self-Test Exercises, Assignments, Quizzes and Exams)
- The Phase of Software Development
  - Basic design strategy
    - four steps- S, D, I, T
  - Pre-conditions and post-conditions
    - assert
  - Running time analysis
    - big-O notation

Self-Test Exercises: 3-6, 11-15, 17-20

Chapter 2

A Review of C++ Classes (Lecture 2)

- OOP, ADTs and Classes
- Class Definition, Implementation and Use
- Constructors and Value Semantics

More on Classes (Lecture 3)

- Namespace and Documentation
  - three ways to use namespace; pre- /post-conditions
- Classes and Parameters
  - value, reference, const reference
- Operator Overloading
  - nonmember, member and friend function

Self-Test Exercises: 1, 4 ,513,15,17,21,23, 25,28,31

Chapter 3

- A container class is a class that can hold a collection of items.
- Container classes can be implemented with a C++ class.
- The class is implemented with
  - a header file (containing documentation and the class definition) bag1.h and
  - an implementation file (containing the implementations of the member functions) bag1.cpp.
- Other details are given in Section 3.1, which you should read, especially the real bag code.

Time Analysis of the Bag Class

- count – the number of occurrence
- erase_one – remove one from the bag
- erase – remove all
- += - append
- b1+b2 - union
- insert – add one item
- size – number of items in the bag

The Invariant of a Class

- Two rules for our bag implementation
  - The number of items in the bag is stored in the member variable used;
  - For an empty bag, we don't care what is stored in any of data; for a non-empty bag, the items are stored in data[0] through data[used-1], and we don't care what are stored in the rest of data.
- The rules that dictate how the member variables of a (bag) class are used to represent a value (such as a bag of items) are called the invariant of the class

What's the most important, then?

- Concept of Container Classes
  - the bag class is not particularly important
- Other kinds of container classes
  - Other types of bags using typedef
  - sequence – similar to a bag, both contain a bunch of items. But unlike a bag, the items in a sequence is arranged in order.(assignment 2)
- Self-Test Exercises: 1,3, 5,10,11,14,18-24

Chapter 4

- Pointers
  - *(asterisk) and &(ampersand) operators
- Dynamic Variables and new Operator
  - Dynamic Arrays and Dynamic Objects
  - Stack (local) vs. heap (dynamic) memory
- Garbage Collection and delete Operator
- Parameters revisited
  - Pointers and Arrays as Parameters

Why Dynamic Classes

- Limitation of our bag class
  - bag::CAPACITY constant determines the capacity of every bag
  - wasteful and hard to reuse
- Solution:
  - provide control over size in running time, by
  - pointers and dynamic memory
  - => dynamic arrays
  - => dynamic classes

Dynamic Classes (Ch 4.3–4)

- Pointers Member Variables
- Dynamic Memory Allocation
  - where and how
- Value Semantics (with dynamic memory)
  - assignment operator overloading
  - your own copy constructor
- Destructor
- the Law of the Big Three

Self-Test Exercises:1-4, **16 - 23**, 26- 32

Time Analysis of the Bag Class

- count – the number of occurrence
- erase_one – remove one from the bag
- erase – remove all
- += - append
- b1+b2 - union
- insert – add one item
- size – number of items in the bag

Chapter 5: Linked Lists

- Node Class (Ex 1-4, 6-9)
- Linked List Toolkit (Ex 10 -16)
- The bag Class with a Linked List (Ex 19-21, 23,24,26)
- The sequence class with a Linked List (Ex 27-30: please refer to assignment 4)
- Comparison of dynamic arrays, linked lists and doubly linked lists (Ex 31-36)

Chapter 6: Templates, Iterators and STL

- Template Functions and Template Classes
  - for code that is meant be reused in a variety of settings in a single program
  - Sections 6.1-6.2, Ex 1-3, 6-11
- Iterators (Sec. 6.5- 6.7, Ex 17-28)
  - step through all items of a container in a standard manner
- Standard Template Library (Section 6.3, Ex. 12-16)
  - the ANSI/ISO C++ Standard provides a variety of container classes in the STL

All you need to know about Templates

- Template Function (Ex. 1-3)
  - a template prefix before the function implementation
  - template <class Item1, class Item2, ...>
- Function Prototype
  - a template prefix before the function prototypes
- Template Class (Ex. 6-10)
  - a template prefix right before the class definition
- Instantiation (Ex 11)
  - template functions/classes are instantiated when used

Better Understanding of classes and functions

Iterators (Sec. 6.5- 6.7, Ex 17-28)

- We discussed how to build an iterator for the linked list
- so that each of the containers can build its own iterator(s) easily
- A node iterator is an object of the node_iterator class, and can step through the nodes of the linked list

Linked List Version the bag Template Class with an Iterator

- Most of the implementation of this new bag is a straightforward translation of the bag in Chapter 5 that used an ordinary linked list
- Two new features
  - Template class with a underlying type Item
  - iterator and const_iterator – defined from node_iterator and const_node_iterator, but use the C++ standard [...) left inclusive pattern

Standard Template Library (STL)

- The ANSI/ISO C++ Standard provides a variety of container classes in the STL
  - set, multiset, stack, queue, string, vector
- Featured templates and iterators
- For example, the multiset template class is similar to our bag template class
- More classes summarized in Appendix H

Chapters 7/8 Stacks and Queues

- Stacks and LIFO (Read Chapter 7, esp. 7.1 and 7.3)
  - Self-Test: 1, 2, 7, 8, 9, 10,
- Queues and FIFO (Read Chapter 8, esp. 8.1 and 8.3)
  - Self-Test: 1, 2, 6,10, 11,12
- Priority Queues (Read Section 8.4)
  - Self-Test: 16, 17
- References Return Values (Read Section 8.5 and p. 302 in Chapter 6)
  - Self-Test: class note of Lecture 12

Chapter 9 Recursive Thinking

- Recursive Functions (Section 9.1)
  - Recursive Calls and Stopping Cases
  - Activation Record and Run-Time Stack
  - Self-Test: Exercises 1-4
- Reasoning about Recursion (Section 9.3)
  - Infinite Recursion and One Level Recursion
  - Ensure no Infinite recursion and Correctness
  - Ex. 9, 10 on page 448
- Applications of Recursion (Optional :Section 9.2)

Midterm

Two parts (100 minutes, 30 questions)

- Short Answers (10), e.g.

  What is an automatic default constructor, and what does it do?

- Multiple Choices (20), e.g.

Suppose that the foo class does not have an overloaded assignment operator. What happens when an assignment a=b; is given for two foo objects?

A. The automatic assignment operator is used

B. The copy constructor is used

C. Compiler error

D. Run-time error