Study Guide to Exam 2

This exam covers the entire course, but emphasis will be on topics covered since exam 1 (see Study Guide to Exam 1). You are reponsible for the contents of lectures 1–21, 22a, 22b, and as much of lectures 23a and 23b as we are able to cover in class on November 29. You are also responsible for the corresponding class demos, concepts used in the problem sets, and the entire textbook (Chapters 1–18) except for the following chapeters and sections:

- Omit section 8.4 (event traces)
- Omit sections 9.3 (ragged arrays), 9.5 (`StringArray`), 9.6 (hashing), and 9.7 (dictionary example)
- Omit chapter 11 (modules and makefiles)
- Omit section 16.6 (virtual inheritance)

Below is an index to the lecture notes. It lists all of the sections, subsections, and slide titles from all of the lectures.

1 About This Course [lecture 01]

- Where to find information
- Course mechanics
- Topics to be Covered
- Course goals - practical
- Course goals - conceptual

2 Kinds of Programming [lecture 01]

- Two views of programming
- Problem solving
- Software Construction
- Programming in the large

3 C++ Programming Standards [lecture 01]

- Three commandments for this course
- Can is not the same as should!

4 C++ Overview [lecture 02]

4.1 C++ Goals

- Why did C need a ++?
- C++ was Designed for Modeling
- General properties of C++

4.2 Comparison of C and C++

- C++ Extends C
- Some Extensions in C++

4.3 Tools

- Tools
- Recommended IDE's

5 Example [lecture 02]

5.1 Insertion sort

- Generic Insertion Sort

5.2 C version

- C version
  ee code demo02

5.3 C++ version

- C++ version
  ee code demo02

6 Example [lecture 03]

- C++ version
  ee code demo03

6.1 Header file

- `dataPack.hpp`
- class DataPack
- Class elements
- Inline functions
- Visibility
- Constructor
- Constructor
- Destructor
- Destructor

6.2 Implementation File

- `dataPack.cpp`
- File I/O

6.3 Main Program

- `main.cpp`

7 Building Your Code [lecture 03]

- Manual compiling and linking
- Makefile
- Integrated Development Environment (e.g., Eclipse)
- Integrated Development Environment (e.g., Eclipse)
- Integrated Development Environment (e.g., Eclipse)

8 Remarks on Laboratory Work [lecture 04]

- Toolset to use for course work
- Working remotely
- 1. Replicate the Zoo environment on your own machine
- 2. Remote login to the Zoo
- 3. Set up a virtual Zoo desktop on your machine
- Homework submission

9 Review and Readings [lecture 04]

- A brief course review to date
- How to use the textbook

10 A Survival Guide for PS1 [lecture 04]

- Operator extensions
- Adding new methods
- Two kinds of functions
- An ambiguity with operator extensions
- Operator call example: Top-level function
- Operator call example: Member function
- Back to PS1

11 More on C++ I/O [lecture 04]

- Opening and closing streams
- Reading data
- Writing data
- Manipulators
- End of file and error handling

12 Functions and Methods [lecture 05]

12.1 Parameters

- Call by value
- Call by pointer
- Call by reference
- I/O uses reference parameters

12.2 Choosing Parameter Types

- How should one choose the parameter type?
- Sending data to a function: call by value
- Sending data to a function: call by reference or pointer
- Receiving data from a function

12.3 The Implicit Argument

- The implicit argument
- `this`

13 Simple Variables [lecture 05]

- L-values and R-values
- Simple variable declaration
- Simple assignment
- Automatic dereferencing

14 Pointers [lecture 05]

- Pointer values
- Pointer creation
- Pointer variables
- Pointer assignment
- Following a pointer
- Pointer example
- Pointer declaration syntax

15 References [lecture 05]

- Reference types
- Reference declarators
- Use of named references
- Reference parameters
- Reference return values
- Custom subscripting
- Constant references
- Comparison of reference and pointer

16 IO Demos [lecture 07]

- Handling data errors and end of file
- How to write a test program

17 Introduction to Classes [lecture 07]

- Classes, visibility, functions, inline

18 BarGraph Demo [lecture 07]

- Bar Graph Demo

18.1 Specification

- Bar graph sample input and output
- Bar graph data structure
- UML Diagram

18.2 `graph.hpp`

- Notes: `graph.hpp`

18.3 `graph.cpp`

- Notes: `graph.cpp`

18.4 `row.hpp`

- Notes: `row.hpp`

18.5 `row.cpp`

- Notes: `row.cpp`

18.6 `rowNest.hpp`

- Nested classes: `rowNest.hpp`

19 Storage Managemet [lecture 08]

- Variables and storage
- Example of a variable
- Properties of variables
- Storage classes
- Assignment and copying
- Static data members
- Static function members
- Five common kinds of failures

20 Bells and Whistles [lecture 08]

- Optional parameters
- `const`
- `const` implicit argument
- Operator extensions

21 Classes [lecture 08]

- What is a class?

22 Derivation [lecture 09]

- Class relationships
- What is derivation?
- Instances
- Some uses of derivation
- Example: Parallelogram
- Example: Rectangle
- Example: Square
- Notes on `Square`

23 Construction, Initialization, and Destruction [lecture 09]

- Structure of an object
- Example of object of a derived class
- Referencing a composed object
- Referencing a base object
- Initializing an object
- Construction rules
- Destruction rules
- Constructor ctors
- Initialization ctors
- Initialization not same as assignment
- Copy constructors

24 Polymorphic Derivation [lecture 09]

- Polymorphism and Type Hierarchies
- Polymorphic pointers
- Virtual functions
- Unions and type tags
- Virtual destructors

25 Polymorphic Derivation (cont.) [lecture 10]

- Uses of polymorphism
- Multiple representations
- Heterogeneous containers
- Run-time variability
- Pure virtual functions
- Abstract classes

26 Name Visibility [lecture 10]

- Private derivation (default)
- Private derivation example
- Public derivation
- Public derivation example
- The `protected` keyword
- Protected derivation
- Privacy summary

27 Name Visibility Revisited [lecture 10]

- Surprising example 1
- Surprising example 2: contrast the following
- Surprising example 3

28 Name Visibility Revisited [lecture 11]

- Names, Members, and Contexts
- Declaration and reference contexts
- Declaration context example
- Reference context example
- Inside and outside class references
- Examples
- Inherited names
- Inheritance example
- Inaccessible base class

29 Interacting Classes and UML [lecture 11]

- What is a Class: Syntax
- Class Relationships
- Class Relationship Between Two Classes
- Class `B` appears in Definition of Class `A`
- `B` as Data Members in `A`
- `B` as Data Members in `A`
- Creation and Deletion
- Example: `BarGraph` Class Interaction

30 Interacting Classes and UML (continued) [lecture 12]

- Association Relationship
- Accessing `B` in `A`'s methods
- "Law" of Consistency/Encapsulation
- Limiting coupling between classes
- "Law" of Demeter

31 Review for Exam 1 [lecture 12]

- Goals of OO Programming
- Insertion sort example
- Compiling and linking
- Compiler errors
- Tool set
- `C++` goodies
- Stream I/O
- Functions and methods
- Variables and data
- BarGraph demo
- More on variables
- Five kinds of memory errors
- `C++` bells and whistles
- Derivation
- Polymorphic derivation

32 Privacy Revisited (again) [lecture 13]

- Visibility rules
- Explicit privacy attributes
- Implicit privacy attributes
- Implicit privacy chart
- Summary

33 Problem Set 2 Code Review [lecture 13]

- A retrospective look at PS2
- Testing

34 More on Course Goals [lecture 14]

- Low-level details
- Example picky detail
- Efficient use of resources
- Efficiency measurement

35 Demo: Stopwatch [lecture 14]

- How to measure run time of a program
- High resolution clocks
- Measuring time in real systems
- Realtime measurements
- `HirezTime` class
- Versions of `HirezTime`
- `HirezTime` structure
- Printing a `HirezTime` number
- `StopWatch` class
- Casting a `StopWatch` to a `HirezTime`
- Why it works

36 Demo: Hangman Game [lecture 14]

- Hangman game

37 Runtime Tester [lecture 15]

- Modularizing timing tests
- Structure of `class Tester`
- Objective **??**
- Objective **??**
- Objective **??**
- Objective **??**
- Member function pointers
- Declaring member function pointers
- Using `typedef` with member function pointers
- Creating member function pointers
- Using member function pointers

38 Demo: Hangman Game [lecture 15]

- Hangman game
- Overall design
- Use cases
- Code structure: Model
- Code structure: Viewer and controller
- Class `Game`
- Storage management
- String store

39 Demo: Hangman Game (continued) [lecture 16]

- Refactored hangman game
- Flex arrays
- Flex array implementation issues
- String store limitation
- Refactoring `Board` class

40 Templates [lecture 16]

- Template overview
- Template functions
- Specialization
- Template classes
- Compilation issues
- Template parameters
- Using template classes

41 The C++ Standard Library [lecture 17]

- A bit of history
- Containers
- Common container operations
- `vector<T>`
- Iterators
- Iterator example
- Using iterator inside a class
- Using subscripts and `size()`
- Algorithms
- STL `sort` algorithm
- Reverse sort example
- Reverse sort example (cont.)
- `pair<T1, T2>`
- `map<Key,Val>`
- Using a `map<Key,Val>`
- Copying from one container to another
- Copying from `map` to vector of pairs
- `string` class

42 Casts and Conversions [lecture 18]

- Casts in `C`
- Different kinds of casts
- `C++` casts
- Explicit cast syntax
- Implicit casts
- Ambiguity

- `explicit` keyword

43 Operator Extensions [lecture 18]

- How to define operator extensions
- Other special cases

44 Virtue Demo [lecture 18]

- Virtual virtue
- Main virtue

45 Linear Data Structure Demo [lecture 18]

- Using polymorphism
- Interface file
- Class Linear
- Example: Stack
- Example: Queue
- Class structure
- C++ features
- `#include` structure

46 Exceptions [lecture 19]

- Exceptions
- Exception handling
- C-style solution using status returns
- C++ exception mechanism
- Throwing an exception
- Catching an exception
- What kind of object should an exception throw?
- Standard exception class
- Catching standard exceptions
- Deriving your own exception classes from `std::exception`
- Multiple catch blocks
- Rethrow
- A subtle fact about rethrow
- Example
- Results
- Throw restrictions
- Uncaught exceptions: Ariane 5
- Uncaught exceptions: Ariane 5 (cont.)
- Termination

47 Ordered Container [lecture 20]

- Demo `20a-Multiple`
- `Ordered` base class
- `Container` base class
- `class Item`
- `class Linear`
- `class PQueue`

48 Multiple Inheritance [lecture 20]

- What is multiple inheritance
- Object structure
- Diamond pattern

49 Handling Circularly Dependent Classes [lecture 20]

- Tightly coupled classes
- Example: `List` and `Cell`
- Circularity with `#include`
- What happens?
- Resolving circular dependencies

50 Template Example [lecture 20]

- Using templates with polymorphic derivation
- Container class hierarchy
- Item class hierarchy
- `Ordered` template class
- Alternative `Ordered` interfaces

51 Linear Container Design [lecture 21]

- Overview of linear container example
- Differences in functionality
- Class structure
- Template structure
- Further extensions
- Two problems
- Defining `KeyType`
- Constructing the data elements
- `21a-Multiple-template`
- Storage management

52 STL and Polymorphism [lecture 21]

- Derivation from STL containers
- Replacing authority with understanding
- Two kinds of derivation
- How are they the same?
- What is simple derivation good for?
- What are the problems with simple derivation?
- What is polymorphic derivation good for?
- What are the problems of polymorphic derivation?
- Contrasts between simple and polymorphic derivation
- Containment as an alternative to simple derivation
- Argument for containment
- STL container as a base class
- Can I turn an STL container into a polymorphic base class?
- A polymorphic base class
- Dynamic cast

53 Design Patterns [lecture 22a]

- General OO principles
- What is a design pattern?
- Adaptor pattern
- Adaptor diagram
- Indirection
- Proxy pattern
- Polymorphism pattern
- Polymorphism diagram
- Controller
- Three kinds of controllers
- Bridge pattern
- Bridge diagram
- Subject-Observer or Publish-Subscribe: problem
- Subject-Observer or Publish-Subscribe: pattern
- Subject-Observer or Publish-Subscribe: diagram
- Singleton pattern
- `StringStore` example

54 Reusability, Flexibility, and Maintainability [lecture 22b]

- The Waterfall Software Process
- Why a Pure Waterfall Process is Usually Not Practical
- The Spiral Process
- Advantage of OO Design
- Aspect of Reusability
- Making a Class Re-usable
- Reducing Dependency Among Classes

- Aspect of Flexibility
- Some Techniques to Achieve Flexibility
- Roadmap
- What is a Design Pattern
- UML/OMT Notation

55 Graphical User Interfaces [lecture 23a]

- User Interfaces
- Interfaces for `C++`
- Overall Structure of a GUI
- Concurrency and Events
- Event Loop
- A GUI event structure
- Interface between user and system code
- Binding system calls to user functions
- Polymorphic binding
- Binding through callback registration
- Callback using function pointers: GUI side
- Callback using function pointer: User side
- Type safety
- Signals and slots

56 The `gtkmm` Framework [lecture 23a]

- Structure of `gtkmm`
- Compiling a `gtkmm` program
- Linking a `gtkmm` program
- Using a GUI
- Example: `clock`
- Main program

57 Example: Duck Game [lecture 23b]

- Initial Design
- Design Change: add fly()
- Problem
- Anticipating Changes
- Handling Varying Behaviors
- Design
- Programming to implementation vs interface/supertype
- Implementation
- Exercise: Add rocket-powered flying?

58 The Strategy Pattern [lecture 23b]

- Exercise
- Summary: Design Principles

59 Example: KitchenViewer Interface [lecture 23b]

- KitchenViewer Example
- Selecting Antique Style
- KitchenViewer Using Standard Inheritance
- The Abstract Factory Idea
- Abstract Factory Design Pattern Applied to KitchenViewer

60 Abstract Factory Design Pattern [lecture 23b]

- Concrete and Abstract Layers
- Abstract Factory Application Sequence Diagram
- Potential use of this Design Pattern?

61 References [lecture 23b]

62 Example: Starbuzz Coffee [lecture 23b]

- Problem
- Starbuzz UML
- Attempt 1
- Potential Changes
- Design idea
- Design approach 1

63 Decorator design [lecture 23b]

- Decoration Delegation Process
- Decorator Class Model
- Sequence Diagram for Decorator
- Decoration Features
- Exercise: different sizes for beverages

64 Some Common Design Patterns [lecture 23b]

- Example: Weather-O-Rama
- Weather-O-Rama
- Weather-O-Rama Interface
- First Implementation

65 Observer Pattern [lecture 23b]

- Observer Design Pattern
- How does Observer apply these design principles?
- Discussion