Study Guide for CPSC 427/527, Fall 2016

Michael J. Fischer

December 12, 2016

This study guide lists all section titles and slide titles from the posted lecture notes. Numbers in parentheses following bullet points are active links to the lecture notes where the slide appears.

1 About This Course

- Where to find information (01)
- Course mechanics (01)
- Course goals (01)
- *Who* programs and why? (01)
- *How long* does a program last? (01)
- *What* are the characteristics of a good program? (01)
- *When* do good programs matter? (01)
- Important properties for lifecycle management (01)
- *Why* does C++ help one write good programs? (01)

2 Topics to be Covered

- Major Areas (01)
- Course goals - practical (01)
- Course goals - conceptual (01)

3 Kinds of Programming

- Problem solving (01)
- Software Construction (01)
- Industrial-Strength Sofware (01)

4 Why C++?

- C/C++ are popular (01)
- C/C++ is flexible (01)
- Advantages and disadvantages of C++ (01)
- Downsides of C++ (01)

5 C++ Programming Standards

- Five commandments for this course (01)
- Can is not the same as should! (01)
- Rules for preparing your work (01)

- Rules for submitting your work (01)

6 Task List

- Tasks for this week (02)

7 C++ Overview

7.1 C++ Language Design Goals

- Why did C need a ++? (02)
- C++ was Designed for Modeling (02)
- General properties of C++ (02)

7.2 Comparison of C and C++

- C++ Extends C (02)
- Some Extensions in C++ (02)

8 Building a Project

8.1 C/C++ Compilation Model

- What is a project? (02)
- C++ compilation model (02)
- Header files (02)
- What's in a header file? (02)
- What's in an implementation file? (02)
- Compiling in linux (02)
- Linking (02)
- System libraries (02)
- One-line compilation (02)

8.2 Project management

- The job of the project manager (02)
- Command line development tools (02)

8.3 A sample project

- Parts of a simple project (02)
- Dependencies (02)
- A sample Makefile (02)
- Parts of a Makefile (02)
- Macros (02)
- Rules (02)
- Rules (02)
- Dependencies (02)
- Implicit rules (02)

9 Integrated Development Environments

- Graphical development tools: IDEs (02)
- Recommended IDE's (02)
- Integrated Development Environment (e.g., Eclipse) (02)
- Integrated Development Environment (e.g., Eclipse) (02)
- Integrated Development Environment (02)

10 Submission Instructions

- Submitting your assignments (02)

11 Insertion Sort Example

11.1 Program specification

- Design process: Insertion Sort (03)
- A more refined specification (03)

11.2 Monolithic solution

- A first solution (03)
- What is wrong with this? (03)

11.3 Modular solution in C

- A modular solution (03)
- A modular solution (cont.) (03)

11.4 Modular solution in C++

- C++ version (03)

12 Classes

12.1 Header file

- Header file format (03)
- Class declaration (03)
- class DataPack (03)
- Class elements (03)
- Inline functions (03)
- Visibility (03)
- Constructor (03)
- Constructor (03)
- Destructor (03)
- Destructor (03)

12.2 Implementation file

- `dataPack.cpp` (03)
- File I/O (03)

12.3 Main program

- `main.cpp` (03)

12.4 Building `InsertionSortCpp`

- Manual compiling and linking (03)
- Compiling and linking using `make` (03)

13 Problem Set 2 Preview

- Think-A-Dot (04)

14 C++ I/O

- Streams (04)
- Opening and closing streams (04)
- Reading data (04)
- Writing data (04)
- Manipulators (04)
- Print methods in new classes (04)
- Extending the I/O operators (04)
- Remarks on operator extensions (04)
- Why `<<` returns a stream reference (04)
- Must it be inline? (04)

15 End of File and I/O Errors

- Status bits (04)
- Status functions (04)
- What `eof` means (04)
- When `eof` is turned on (04)
- Reading an `int` (04)
- Reading an `int` (cont.) (04)
- Examples (04)
- Common file-reading mistakes (04)
- Failing to read the last number (04)
- Reading the last number twice (04)
- How to read all numbers in a file (04)

16 Functions and Methods

- Call by value (04)
- Call by pointer (04)

17 Functions and Methods

17.1 Parameters

- Call by value (recall) (05)
- Call by pointer (recall) (05)
- Call by reference (05)

- I/O uses reference parameters (05)

17.2 Choosing Parameter Types

- How should one choose the parameter type? (05)
- Sending data to a function: call by value (05)
- Sending data to a function: call by reference or pointer (05)
- Receiving data from a function (05)

17.3 The Implicit Argument

- The implicit argument (05)
- `this` (05)

18 Derivation

- Class relationships (05)
- What is derivation? (05)
- Instances (05)
- Some uses of derivation (05)
- Example: Parallelogram (05)
- Example: Rectangle (05)
- Example: Square (05)
- Notes on `Square` (05)

19 Construction, Initialization, and Destruction

- Structure of an object (05)
- Example of object of a derived class (05)
- Referencing a composed object (05)
- Referencing a base object (05)
- Initializing an object (05)
- Construction rules (05)
- Destruction rules (05)
- Constructor ctors (05)
- Initialization ctors (05)
- Initialization not same as assignment (05)
- Copy constructors (05)
- Move constructors (05)

20 Brackets Example

- Code demo (06)
- The problem (06)
- A bracket matching algorithm (06)
- Program design (06)
- Token class (06)
- Token class (cont.) (06)
- Token design questions (06)
- Token design questions (cont.) (06)
- Stack class (06)
- Stack design questions (06)

- Brackets class (07)
- Brackets design questions (07)
- Main file (07)

21 Storage Management

- Objects and storage (07)
- Name (07)
- Type of a storage object (07)
- Storage block (07)
- Connecting names to objects (07)
- Lifetime (07)
- Storage class (07)
- Dynamic extensions (07)
- Copying (07)
- The double-delete problem (07)
- When does copying occur? (07)
- Assignment (07)
- Copy constructor (07)
- Redefining assignment and the copy constructor (07)
- Static data members (08)
- Example from brackets demo (08)
- Static data member example (08)
- Class definition (08)
- Static function members (08)
- Static class functions are global (08)
- Debugging memory management errors (08)
- Five common kinds of failures (08)
- Memory leak (08)
- Amnesia (08)
- Segmentation fault (08)
- Bus error (08)
- Waiting for eternity (08)
- What kinds of program errors cause these problems? (08)
- What makes these problems hard to diagnose? (08)

22 Bar Graph Demo

- Overview of bar graph demo (08)
- Overview (cont.) (08)
- Method (08)
- Analysis of 08-BarGraph demo (08)
- Analysis of 08-BarGraph demo (09)
- `main.cpp` (09)
- Design issues for `main.cpp` (09)
- `graph.hpp` (09)
- `graph.cpp` (09)
- Class discussion on bargraph design issues (09)
- `graph.cpp` (10)
- Design issues for `Graph` class (10)
- `row.hpp` (10)
- `row.cpp` (10)
- `rowNest.hpp` (10)

23 Introduction to the C++ standard library

24 Handling Circularly Dependent Classes

25 References

26 Uses of Pointers

27 Custody of Objects

- Custody of dynamic extensions (12)
- Move versus copy (12)

28 Move Semantics

- When to move? (12)
- Temporaries (12)
- Move semantics (12)
- Motivation (12)
- Implementation in C++ (12)
- Move and copy constructors and assignment operators (12)
- Default constructors and assignment operators (12)
- Moving from a temporary object (12)
- Forcing a move from a non-temporary object (12)
- The full story (12)
- 12-BracketsWithMove demo (12)

29 Move Demo

- Special member functions demo (13)
- Special member functions demo (13)
- Default constructor and destructor (13)
- Additional constructor (13)
- Copy constructor and move constructor (13)
- Copy assignment (13)
- Move assignment (13)
- Invoking the special functions (13)
- Invoking the special functions (13)
- Invoking the special functions (13)

30 Bells and Whistles

- Optional parameters (13)
- `const` (13)
- `const` implicit argument (13)
- Operator extensions (13)

31 The Many Uses of Classes

- What is a class? (13)

32 Feedback on Programming Style

- Coding Hints (14)
- Zero-tolerance for compiler warnings (14)
- Declaration order in classes (14)
- Construct semantically consistent objects (14)
- Use `break` (14)
- Use `tolower()` (14)
- Use `switch` (14)
- Use stream input to read data (14)

33 Smart Pointer Demo

34 More on Course Goals

35 Clocks and Time Measurement

36 Demo: Stopwatch

37 Storage Model Revisited

38 Functions Revisited

39 Polymorphic Derivation

40 Demo: Craps Game

41 Name Visibility

42 Linear Data Structure Demo

43 Templates

44 Casts and Conversions

45 Operator Extensions

46 Virtue Demo

- Virtual virtue (21)
- Main virtue (21)

47 Exceptions

- Exceptions (21)
- Exception handling (21)
- C-style solution using status returns (21)
- C++ exception mechanism (21)
- Throwing an exception (demo `21b-Exceptions`) (21)
- Catching an exception (21)
- What kind of object should an exception throw? (21)
- Example: Stack template throws exception (21)
- Polymorphic exception classes (21)
- Standard exception class (22)
- Catching standard exceptions (22)
- Deriving your own exception classes from `std::exception` (22)
- Multiple catch blocks (22)
- Rethrow (22)
- A subtle fact about rethrow (22)
- Example of rethrowing an exception (demo 22a-Exceptions-rethrow) (22)
- Results (22)
- Throw restrictions (22)
- Uncaught exceptions: Ariane 5 (22)
- Uncaught exceptions: Ariane 5 (cont.) (22)
- Termination (22)

48 Code Reuse

- Reusable code (22)
- Problems with reusing code (22)
- How to allow variability (22)
- Specifying constraints (22)

49 Linear Containers

- Demo `19-Virtual` (22)
- Sharing in `19-Virtual` (22)
- Abstract containers (22)

50 Ordered Containers

- Demo `22b-Multiple` (22)
- `Ordered` base class (22)
- `class Item` (22)
- `Container` base class (22)
- Additions to `Linear` (22)
- `class Linear` (22)
- `class PQueue` (22)

51 Multiple Inheritance

- What is multiple inheritance (22)
- Object structure (22)
- Diamond pattern (22)
- Virtual derivation (22)

52 Template Example

- Using templates with polymorphic derivation (22)
- Container class hierarchy (22)
- Item class hierarchy (22)
- `Ordered` template class (22)
- Alternative `Ordered` interfaces (22)

53 Linear Container Design

- Overview of linear container example (23)
- `Item` definitions (23)
- Differences in functionality (23)
- Class structure (23)
- Template structure (23)
- Further extensions (23)
- Two problems (23)
- Defining `KeyType` (23)
- Constructing the data elements (23)
- `23a-Multiple-template` (23)
- Storage management (23)

54 STL and Polymorphism

- Derivation from STL containers (23)
- Replacing authority with understanding (23)
- Two kinds of derivation (23)
- How are they the same? (23)
- What is simple derivation good for? (23)
- What are the problems with simple derivation? (23)
- What is polymorphic derivation good for? (23)
- What are the problems of polymorphic derivation? (23)
- Contrasts between simple and polymorphic derivation (23)
- Containment as an alternative to simple derivation (23)
- Argument for containment (23)
- STL container as a base class (23)
- Can I turn an STL container into a polymorphic base class? (23)
- A polymorphic base class (23)
- Dynamic cast (23)

55 Design Patterns

- General OO principles (23)

56 General OO Principles

- General OO principles (24)
- Basic Principles for OO Design (24)
- Low coupling (24)
- High cohesion (24)
- Don't talk to strangers! (24)
- Polymorphism (24)
- Chain of Responsibility (24)

57 Design Patterns

- What is a design pattern? (24)
- Adaptor pattern (24)
- Adaptor diagram (24)
- Indirection (24)
- Proxy pattern (24)
- Polymorphism pattern (24)
- Polymorphism diagram (24)
- Controller (24)
- Three kinds of controllers (24)
- Bridge pattern (24)
- Bridge diagram (24)
- Subject-Observer or Publish-Subscribe: problem (24)
- Subject-Observer or Publish-Subscribe: pattern (24)
- Subject-Observer or Publish-Subscribe: diagram (24)
- Singleton pattern (24)
- `StringStore` example (24)

58 Function-Like Constructs

- Named functions (25)
- Function calls (25)
- Function types (25)
- `typedef` for function types (25)
- A new way to give names to types (25)
- Function pointers (25)
- Anonymous functions (a.k.a. lambda functions) (25)
- Functors (25)
- Closures (25)
- Lambda capture (25)
- Closure example (25)

59 Graphical User Interfaces

- User Interfaces (25)
- Interfaces for C++ (25)
- Overall Structure of a GUI (25)
- Concurrency and Events (25)
- Event Loop (25)
- A GUI event structure (25)
- Interface between user and system code (25)

- Binding system calls to user functions (25)
- Polymorphic binding (25)
- Binding through callback registration (25)
- Callback using function pointers: GUI side (25)
- Callback using function pointer: User side (25)
- Type safety (25)
- Signals and slots (25)

60 The `gtkmm` Framework

- Structure of `gtkmm` (25)
- Compiling a `gtkmm` program (25)
- `> pkg-config gtkmm-3.0 --cflags` (25)
- Linking a `gtkmm` program (25)
- Using a GUI (25)
- Example: `clock` (25)
- Main program (25)