COMP 211-002 (Bakita)
Midterm 2 ~~Study Guide~~ Post-Exam Review Guide
UNC Chapel Hill
Spring 2023

## Overview

For this second third of the course, we finished covering *The C Programming Language* and moved on to discussing what happens under the covers. This exam is not designed to be cumulative, but the topics will be easier to understand with a foundational understanding of C. This exam will be formatted similarly to Midterm 1, containing mostly multiple-choice questions.

## Logistics

As stated in the syllabus, a double-sided sheet of letter paper and a paper copy of *The C Programming Language* are allowed for reference.

Additionally, for this exam, you are allowed a calculator (as some of you asked if you could bring one), however questions will not require or expect it.

Seats will be assigned, with seat assignments posted at the front of the classroom. Arrive early to get your seat assignment and exam—this will allow you to start promptly at 2 PM.

The format of the exam will be very similar to Midterm 1, and will be composed mostly of multiple-choice questions.

For those with university-approved extended time accommodations from ARS, you are welcome to take the exam at ARS, or in SN314 (the office hours room) as proctored by Andrew and Kangda from the LA/TA team.

**Reminder:** Per the syllabus, "Makeup exams will not be offered except in cases of University Approved Absences (UAAs) and final exam exceptions. Please request such absences as far in advance as possible (such as for a known-in-advance competition or religious observance). See the UAA Office for more details: https://uaao.unc.edu/.

Any requests for other exceptions must be made in-person at the instructor's office (SN311), or over the phone (919-590-6103)."

## Exam Topics

This list of topics follows what we covered in class and in assignments, with each topic or topical section annotated with the associated class or assignment number(s). Please consult the course webpage to resolve a class number to an associated reading.

C Topics:

- Preprocessor (Class 11)
  - How to write and use a `#define` directive that takes parameters?
- Pointers (Class 12+; Assignment 3)
  - What is the syntax for declaring a function pointer?
  - How does one pass a function pointer to a function?

1.1.1

- Given an array, how can I use pointer math and the dereference operator to get the *n*th item?
    - When may I also need to use `sizeof()` to get the size of an array item, as part of the pointer math?
- How do I allocate heap memory with `malloc()`?
- How do I resize a heap allocation with `realloc()`?
- How can I perform the same operation as `->` with `.` and `*`? (Class 17)
- When may I want to sort an array of pointers instead of an array of structs? (Class 18)

- `static` (Class 13)
    - Why does a `static` variable in a function remember its previous value next time that function is called?
    - What does the `static` qualifier do to global variables or functions?
- Multi-file C Programs (Class 13)
    - What are the major steps in building a C file into a binary?
    - What arguments would I pass to `gcc` to compile a multi-file C program?
    - When may a header file be useful?
    - Why would I use the `extern` qualifier on a global variable?
- `const` (Class 13; Assignments)
    - When may we want to add this qualifier to a local variable?
- Local Jumps via `goto` (Class 18)
    - How do I declare a label?
    - How do I jump to a label with `goto`?
- Non-Local Jumps via `longjmp()` (Class 18; Assignment 4)
    - Where will my program resume executing after calling `longjmp()` or `siglongjmp()`?
    - How can I detect that I just jumped to another location in my program with one of the `longjmp()` functions?
- Misc Assignment Content (Assignment 3)
    - What arguments does a comparator function for `qsort()` take?

Computer Organization Basics:

- What are the major components of a computer (disk, CPU, DRAM…), what do they look like, and what do they do? (Class 16–18)
- How far can electrical signals travel in a nanosecond? (Class 16)
- Implications of Physical Distance on Computer & Program Design (Class 17)
    - What is the relationship between how long a single memory fetch takes and how long a single computation (eg. 2+2) takes?
    - Why is most of a processor die dedicated to cache memory?
    - Roughly, how long will each of the following operations take in a modern computer relative to one another (see Table 1.1 from Class 16 reading):
        - A single mathematical operation (once data is available) (eg. 2+2)
        - Fetching a single byte from a CPU cache
        - Fetching a single byte from memory (DRAM)

    - Fetching a single byte from a spinning disk
    - Example: A single math operation takes ~1ns, DRAM is at least 100x slower than a single math operation, and a spinning disk is at least 10,000x slower than DRAM.

Systems Fundamentals:

- Practical Two's Complement (Class 12)
    - What is -1 cast to an unsigned int?
    - What is the maximum and minimum value of a 32-bit unsigned integer (`uint32_t`)?
    - What is the minimum value of a 32-bit unsigned integer (`int32_t`)?
    - How do I compute the maximum size of an arbitrary-size unsigned integer?
    - Given the binary representation of a signed integer, how can I tell if it is positive or negative?

1.2.4

- Virtual Memory (Class 14 and later)
    - How does `mmap()` allow us to access a file without explicitly reading it?
    - When can a page fault occur?
    - What is a segmentation fault and why does it occur?
    - If my program crashed immediately after a page fault, what sort of crash would you expect?
    - What will happen if our program accesses a memory address that it is not allowed to?

1.2.5

- Syscalls
    - What are you calling into when using `fopen()`, `fread()`, or `fwrite()`? (Class 14 & 15)
    - What are the major steps involved in I/O on a modern solid-state drive (SSD)? (Class 15)

1.2.6

- Signals (Class 18)
    - What are some common uses for signals?
    - Why do we need signals?
    - How can I add a signal handler to create a difficult-to-kill program?

1.2.7

- Filesystems (Class 19; Assignment 3)
    - What could be the consequences of storing files smaller than the block size on a file system?
    - What is file system fragmentation?
    - What happens when I delete a file?
    - What is the purpose of an inode?

1.2.8

Performance Topics:

- `time` (Class 14 & 15)
    - How can I use `time` to check how long my program takes to run?
    - What is the difference between the `real`, `user`, and `sys` outputs from `time`?

1.3.1

- Given a basic flamegraph, identify which syscalls are taking the longest. (Class 14 & 15)
- Memory Access Patterns (Class 17)
    - What is *spatial locality*, and how does this affect how we write programs?
    - What is *temporal locality*, and how does this affect how we write programs?

1.3.2

- What are the consequences of a dependent load, and how can you identify one?
- Why might I want to avoid running many very short programs, instead combining the work into one long one? (Class 18)

Systems Tools

- What does it mean if `valgrind` complains, "Conditional jump or move depends on uninitialized value(s)"? (Assignments)
- How can I compile and run a program in `gdb` to figure out what line it's crashing on? (Assignments; Misc Classes)
- How to redirect a file into `stdin`? (Assignment 3)
- How to redirect `stdout` to a file? (Class 18)
- How can I use `*` in bash (the shell) to match multiple files? (Class 13)

1.4.2
1.4.1
1.4.3

Content Not on the Exam (that you might have thought could be):

- How to use `perf`
- Specifics of how signals are implemented
- Specifics about a particular filesystem (eg. size of an inode on ext4)
- How to use `KUtrace`

1.4.4 combined many study guide sections

See anything missing above that you spent a lot of time learning? Let me know, I may have accidentally overlooked it.

## Contents-Ordered Readings Guide

This section of the guide transliterates the *Contents* sections of all the textbooks we have used in the class, annotating all according to the below legend to show what we consider this course to have covered, what we expect as foreknowledge (and will not be explicitly testing), what we will be skipping, and what we will get to later. We have worked hard to keep these readings as brief as possible, and strongly encourage you to both read the text and work through at least some of the included problems. For many of the topics here, further depth or an alternate explanation can be found in videos or further references linked or referenced on the course webpage.

Legend:

| | |
|---|---|
| **Topic** | Assigned reading *since* Midterm 1 (topic considered to be covered). |
| ~~Topic~~ | Assigned reading before Midterm 1. |
| Topic | Assumed from prior classes. |
| *Topic* | Skipped content. |
| Topic | Reading that may be assigned after Midterm 2. |
| * | Contains important reference. |

The C Programming Language (2nd Edition):

- Chapter 1. A Tutorial Introduction
  - ~~1.1 Getting Started~~
  - ~~1.2 Variables and Arithmetic Expressions~~
  - 1.3 The For Statement
  - ~~1.4 Symbolic Constants~~
  - *1.5 Character Input and Output*

        - *See comment on 7.1.*
    - ~~1.6 Arrays~~
    - 1.7 Functions
    - 1.8 Arguments–Call by Value
    - ~~1.9 Character Arrays~~
    - **1.10 External Variables and Scope** 1.1.5
- Chapter 2. Types, Operators, and Expressions
    - ~~2.1 Variable Names~~
    - ~~2.2 Data Types and Sizes~~
    - ~~2.3 Constants~~
    - 2.4 Declarations
    - 2.5 Arithmetic Operators
    - 2.6 Relational and Logical Operators
    - ~~2.7 Type Conversions~~
    - 2.8 Increment and Decrement Operators 1.1.7
    - ~~2.9 Bitwise Operators~~
    - 2.10 Assignment Operators and Expressions
    - ~~2.11 Conditional Expressions~~
    - 2.12 Precedence and Order of Evaluation*****
- Chapter 3. Control Flow
    - 3.1 Statements and Blocks
    - 3.2 If-Else
    - 3.4 Else-If
    - 3.5 Switch
    - 3.6 Loops—While and For
    - 3.7 Loops—Do-while
    - 3.8 Break and Continue
    - **3.9 Goto and Labels**
- Chapter 4. Functions and Program Structure
    - 4.1 Basics of Functions
    - 4.2 Functions Returning Non-integers
    - **4.3 External Variables** 1.1.5
    - **4.4 Scope Rules**
    - **4.5 Header Files**
    - **4.6 Static Variables** 1.1.4, 1.1.7
    - *4.7 Register Variables*
        - *Very uncommonly used, and deprecated in C++.*
    - 4.8 Block Structure
    - 4.9 Initialization
    - 4.10 Recursion
    - ~~The C Preprocessor~~
- Chapter 5. Pointers and Arrays
    - ~~5.1 Pointers and Addresses~~
    - ~~5.2 Pointers and Function Arguments~~

- ~~5.3 Pointers and Arrays~~
- **5.4 Address Arithmetic** 1.1.2
- ~~5.5 Character Pointers and Functions~~
- ~~5.6 Pointer Arrays; Pointers to Pointers~~
- 5.7 Multi-dimensional Arrays
- 5.8 Initialization of Pointer Arrays
- 5.9 Pointers vs. Multi-dimensional Arrays
- ~~5.10 Command-line Arguments~~
- **5.11 Pointers to Functions** 1.1.1
- *5.12 Complicated Declarations*
  - *This can be a helpful reference in case one comes across a cryptic declaration, but is otherwise beyond the scope of this class.*

- Chapter 6. Structures
  - ~~6.1 Basics of Structures~~
  - ~~6.2 Structures and Functions~~
  - ~~6.3 Arrays of Structures~~
  - ~~6.4 Pointers to Structures~~
  - ~~6.5 Self-referential Structures~~
  - *6.6 Table Lookup*
    - *This section only includes a very complicated example. While it would be excellent to understand, it's not essential.*
  - ~~6.7 Typedef~~
  - ~~6.8 Unions~~
  - ~~6.9 Bit-fields~~
- Chapter 7. Input and Output
  - ~~7.1 Standard Input and Output~~
    - *Getchar and putchar are uncommonly used, and not common to other languages. Notes on input redirection are valuable though.*
  - ~~7.2 Formatted Output—Printf~~
  - *7.3 Variable-length Argument Lists*
    - *Rarely used, prone to error, and different from most modern languages.*
  - 7.4 Formatted Input—Scanf
  - ~~7.5 File Access~~
  - ~~7.6 Error Handling—Stderr and Exit~~
  - **7.7 Line Input and Output**
    - **[Assigned in Assignment 3]**
  - 7.8 Miscellaneous Functions
    - …
    - **7.8.5: Storage Management (L11)** 1.1.3
    - …
- *Chapter 8. The UNIX System Interface*
  - *8.1 File Descriptors*
  - *8.2 Low Level I/O—Read and Write*
  - *8.3 Open, Creat, Close, Unlink*

        - *8.4 Random Access—Lseek*
        - *8.5 Example—An Implementation of Fopen and Getc*
        - *8.6 Example—Listing Directories*
        - *8.7 Example—A Storage Allocator*
    - *Appendix A**
    - *Appendix B**
    - *Appendix C*

Computer Systems: A Programmer's Perspective (1st Edition):

- Chapter 2: Representing and Manipulating Information
    - 2.1 Information Storage
        - ~~2.1.1 Hexadecimal Notation~~
        - ~~2.1.2 Words~~
        - ~~2.1.3 Data Sizes~~
        - ~~2.1.4 Addressing and Byte Ordering~~
        - ~~2.1.5 Representing Strings~~
        - ~~2.1.6 Representing Code~~
        - 2.1.7 Boolean Algebras and Rings
            - This should have been covered by your COMP 283/MATH 381 prerequisite. Worth reading if 2.1.8-2.1.10 use terms you don't recognize.
        - ~~2.1.8 Bit-Level Operations in C~~
        - ~~2.1.9 Logical Operations in C~~
        - ~~2.1.10 Shift Operations in C~~
    - 2.2 Integer Representations
        - ~~2.2.1 Integral Data Types~~
        - **2.2.2 Unsigned and Two's-Complement Encodings**
        - **2.2.3 Conversions Between Signed and Unsigned**
        - ~~2.2.4 Signed vs Unsigned in C~~
        - ~~2.2.5 Expanding the Bit Representation of a Number~~
        - ~~2.2.6 Truncating Numbers~~
        - ~~2.2.7 Advice on Signed vs Unsigned~~
    - …
- Chapter 3 Machine-Level Representation of Programs
    - …
    - **3.9 Heterogeneous Data Structures**
        - **3.9.1 Structures**
        - **3.9.2 Unions**
    - …
- Chapter 4 Processor Architecture
- Chapter 5 Optimizing Program Performance
- Chapter 6: The Memory Hierarchy
    - 6.1 Storage Technologies
        - 6.1.1 Random-Access Memory
        - ~~6.1.2 Disk Storage~~

        - 6.1.3 Storage Technology Trends
    - …
- Chapter 7: Linking
    - **7.1 Compiler Drivers**
    - **7.2 Static Linking**
    - …
- Chapter 8: Exceptional Control Flow
    - 8.1 Exceptions
    - **8.2 Processes**
        - ~~8.2.1 Logical Control Flow~~
        - **8.2.2 Private Address Space**
        - **8.2.3 User and Kernel Modes**
        - **8.2.4 Context Switches**
    - **8.3 System Calls and Error Handling**
    - 8.4 Process Control
    - **8.5 Signals**
    - **8.6 Nonlocal Jumps**
    - **8.7 Tools for Manipulating Processes**
    - 8.8 Summary
- Chapter 9: Measuring Program Execution Time
    - **9.1 The Flow of Time on a Computer System**
    - …
- Chapter 10: Virtual Memory
    - …
    - 10.9 Dynamic Memory Allocation
        - **10.9.1 The Malloc and Free Functions**
        - **10.9.2 Why Dynamic Memory Allocation**
        - …
    - …

Operating System Concepts (7th Edition):

- Chapter 1
    - **1.1 What Operating Systems Do**
    - **1.2 Computer-System Organization**
    - …
- …

UNIX Power Tools (1st Edition; -1 from chap. num. for 2nd Ed.):

- …
- Chapter 14:
    - **14.1 Using Standard Input and Output**
    - …
- …

Computer Organization and Design (3rd Edition):

- Chapter 1: Computer Abstractions and Technology
    - …

    - **1.3 Under the Covers** 1.2.1, 2.0.1
    - …
- …
- Chapter 7: Large and Fast: Exploiting the Memory Hierarchy
    - **7.1 Introduction** 2.0.1
    - …
- …

Understanding Software Dynamics (1st Edition):

- Preface
- **Chapter 1: My Program Is Too Slow** 1.3.2
- Chapter 2: Measuring CPUs
- Chapter 3: Measuring Memory
    - **3.1 Memory Timing**
    - **3.2 About Memory**
    - **3.3 Cache Organization**
    - **3.4 Data Alignment**

    1.3.2, 2.0.1
    - …
- …

Changelog

| | |
|---|---|
| PM, March 24 | Initial release, including the Contents-Ordered Readings Guide |
| AM, March 26 | Released *Logistics* section |
| AM, March 26 | Added logistical info for students with extended-time accommodations |
| PM, March 26 | Released *Exam Topics* section; removed construction notice |
| PM, March 26 | Clarified what we're looking for re: operation/memory speeds |
| PM, March 26 | Added more detail to section on extended-time accommodations |
| PM, March 27 | Clarified what we're looking for re: pointer math |

AM, March 3 Released
AM, March 3 Updated