**CS 2060 Programming with C**

**Final Review Chapters 1 – 12**

**Mon/Wed class -- Wednesday Dec 13th 8:00 am – 10:00 am**

**Tues/Thurs class -- Thursday Dec 14th 8:00 am – 10:00 am**

This is an overview of the material you should study for the final exam.

**Chapter 1 - Introduction to Computers, the Internet and the Web**

* Difference between compiler, logical and runtime errors
  + What causes each type of error?
  + When or how they occur

Compiler: missing semicolon

Logical: add instead of multiply

Runtime: memory problems, divide by 0

**Chapter 2 – Introduction to C**

* Know how to print output to console and obtain input from keyboard

Printf(), scanf()

* Know the different data types and what a "data type" tells the compiler
  + int, float, double, char

Type and size of data to store

* Know when and how to declare variables, assign values to, use in statements
* Know when and how to declare constants, assign values to, use in statements
* Be able to use numeric operations (+, - , \*, /, %)
  + Integer division and the issues you can have with integer division

Whole numbers only, numbers get rounded, produce incorrect results

* Know how to evaluate numeric expressions
* Understand operator precedence when it comes to evaluating numeric expressions

(), \* / %, + -, <, <=, >, >=, ==, !=

* Know what the **relational operators** and **equality operators** are

Relational: <, <=, >, >=

Equality: ==, !=

* Know how to use relational operators to create boolean expressions
* Know how to evaluate boolean expressions

**Chapter 3 – Structured Program Development in C**

* Understand how **if statements** work
* Understand how a **while loop** works
* Simple type conversions - explicit casting – we did this with integer division example
* Be able to use the increment and decrement operators
* Be able to understand code and/or write code snippets with **if statements**
* Be able to understand code and/or write code snippets with **while loops**

**Chapter 4 – C Program Control**

* Understand how a ***for loop*** works
* Understand how a ***do-while loop*** works
* Know when to use one loop structure over the other (while, for, do-while)
* Know how loops work when they are nested
* Understand what causes infinite loops
* Understand how a **switch statement** works
  + Understand how fall-through behavior works
  + Understand how the default case works
* Understand char data types
  + What ASCII code is and how it is used at a high level
* Know what the **logical operators** are
* Know how to use **logical operators** with the **relational/equality operators** to create boolean expressions
* Be able to understand code and/or write code snippets with **for loops** and **do-while loops**
* Be able to understand code and/or write code snippets with **switch statements**

**Chapter 5 – C Functions**

* Know how to use these 2 basic math functions - sqrt, pow
* Know how to define and create a function, so understand:
  + Function prototypes
  + Function invocation
  + Function definition
* Understand how to pass values to a function and how to return values from functions
* Understand automatic conversions:
  + Know that conversions occur when argument types do not correspond to parameter types
  + Know what kinds of issues automatic conversions can cause
* Understand the scope of variables and how to determine the scope of variables
* Understand what ***pass by value*** means and the impact on variables when a function is called
* Know what recursion is on a high-level, you will NOT be asked to write a recursive function
* Be able to explain code, trace code, and write code snippets with functions

**Chapter 6 – C Arrays**

* Know how to declare, create, and initialize an array (one or two dimensional)
* Know how to access elements within an array (one or two dimensional)
* Understand initialization of arrays
  + What is in array when you don’t initialize the array
  + Understand initializer lists (i.e. how they work, what if fewer items than array elements, etc.)
* How to manipulate elements within an array (i.e. compute sum, find largest, display elements, etc.)
* Understand what happens with arrays
  + When code accesses elements in an array outside the array bounds
  + When there is an off-by-one situation (mostly issue when forgetting arrays are zero-based)
* Understand character arrays
  + Know how to create and initialize strings using character arrays
  + Understand the importance of the null character in strings
  + How to read and display strings
* Understand and know how to ***pass arrays to functions***
  + Understand what ***pass by reference*** means and the impact when passing an array to a function
  + Know how to pass an entire array or a single element in the array to functions
  + Understand when to use ***const*** qualifier on an array in a parameter list
* Know how to perform a linear search for a key value
* Understand the **concept** of how the binary search works
* Know when one search is better over the other
* Be able to explain code, trace code, and write code snippets with arrays

**Chapter 7 – C Pointers**

* Understand what a pointer variable is
  + How pointers are different from other variables
  + What is the purpose of pointers
* Know how to create a pointer and initialize it
* Know how to use the pointer operators **&** and ***\****
  + Know what each operator does
  + Know when do you use one over the other
  + Understand code that contains these operators
* Understand and know how to ***pass pointers to functions***
  + Understand what ***pass by reference*** means and the impact when passing a pointer to a function
  + Understand and know what needs to be in argument and parameter lists when passing pointers
* Understand the 4 different cases of using the **const** qualifier with pointers
  + Non-constant pointer to non-constant data
  + Constant pointer to non-constant data
  + Non-constant pointer to constant data
  + Constant pointer to constant data
* Know how to use the ***sizeof*** operator
* Know how to use pointers in expressions and how to perform pointer arithmetic
  + Comparing pointers
  + Incrementing and decrementing pointers
  + Adding and subtracting values from pointers
  + Subtracting one pointer from anther
* Understand what a void pointer is
* Understand the relationship between arrays and pointers
* Understand how to create, access, manipulate an array of pointers
* Be able to explain code, trace code, and write code snippets with pointers

The following is the material covered after the midterm. The final will be focused on the following material; however, you need to understand the material from chapters 1 - 7 since it is the foundation for chapters 8-12.

**Chapter 8 – C Characters and Strings**

* Know the difference between characters and strings

Char: ‘ ‘

String: “ “

* Know how to declare and initialize strings
  + Know that strings are terminated with null character
  + Know that array needs to be big enough to store string and null character
  + Know what happens when there is not enough room for null character

Gets left off

* If I ask you to use any of the functions, I will give you the prototypes but look over Lecture #22 notes and understand how to use these and the issues that can occur:
  + fgets
  + strcpy and strncpy
  + strcat and strncat
  + strcmp and strncmp

**Chapter 9 – C Formatted Input/Output**

* Read Lecture #23 notes and book section called “Printing Strings and Characters”
  + Know when to use **%c** or **%s**
* Know what field width and precision specifiers do when included in printf

% + Width.precision + type

Min. for numbers

Max for strings

**Chapter 10 – C Structures, Unions, Bit Manipulation and Enumerations**

* Understand how a structure is different from an array
* Know how to define a structure
* Know how to declare and initialize variables of the structure type
* Know how to access members within a structure
  + Know difference between the dot and arrow operators
* Understand these two notations
  + employeePtr->firstName
    - (\*employeePtr).firstName
* Understand initialization of structures
  + What is in structure when you don’t initialize the structure
  + Understand initializer lists (i.e. how they work, what if fewer items than structure members, etc.)
* How to manipulate members within a structure (i.e. display members, things like assignment #11, etc.)
* Know how to declare and create more complex data structures
  + Structure with an array as a member
  + Structure containing another structure
  + Array of structures
* Know how to ***pass structures to functions***
  + Understand default is ***pass by value***
  + Know how to ***pass by reference*** with a structure
  + Know how to pass an **entire structure** or **individual members in the structure** to functions
  + Know what happens when pass array of structures to a function or passing structure that contains an array
* Understand the difference between structures and unions
* Understand the difference between the bitwise operators and the logical operators
* Know how to use bitwise AND - bitwise OR operators
* Know that the dot operator has a higher precedence than address operator

**Chapter 11 – C File Processing (11.1 – 11.4 – Sequential Files Only)**

* Know how to create and use a file pointer
* Understand the following functions when dealing with files
  + fopen
  + fgets
  + fscanf
  + fprintf
  + feof
  + fclose
* Be able to explain code, trace code, and write code snippets dealing with files

**Chapter 12 – C Data Structures**

* Understand the different between **static** data structures and **dynamic** data structures
* Understand how to create a node that will be used in a linked list - self-referential structure
* Understand dynamic allocation and deallocation of memory for a node
* Know the strengths weaknesses of arrays vs linked lists
* Understand how to connect nodes together when creating a linked list
* Know the differences between linked lists, stacks, queues
* Be able to explain code and trace code snippets with a linked list

**Study tips**

* Review lectures notes
* Review programming examples in lecture notes
* Review homework assignments (assignment #7 and #11 are good ones!)
* Go over case studies in the book - see if you can write the code
* Do the easier exercises in book
* Practice solving problems on paper
* Practice writing little code snippets on paper
* Get enough sleep and drink water - your brain needs water to work properly!