

CS 1044 Summer II, 2000

Exam Review

Chapters 1 and 2

- Understand the difference between machine language, assembly language, and a high level language like C++. What are the advantages of using a high level language? What is the purpose of a compiler?
- Algorithm - What is an algorithm?
- Divide and conquer - How do we use divide and conquer to help solve a problem?

Chapter 3

- Understand the different types of programming errors.
- Understand the difference between syntax and semantics.
- Compiler directives - Why do we use compiler directives, specifically, the "#include" directive?
- Variables and Constants
 - Data Type
 - What are the simple data types?
 - Why do we need to know the data type of a variable or constant?
 - Identifiers
 - What are valid identifiers? What are invalid identifiers?
 - Variables
 - Understand that a variable is actually a location in memory that has a name (identifier) associated with it.
 - What is the difference between a variable and a constant?
 - What is the value of the variable if we declare it, but don't initialize it?
- Arithmetic
 - Understand the different operators (- (negation), *, /, %, +, -). In particular, understand the difference between integer and real division. Understand what happens when we assign a real number to an integer, and what happens when we assign an integer to a real. Make sure you understand the arithmetic from the first homework!
 - Understand the increment and decrement operators. What happens when we use ++x instead of x++?

Chapter 4

- Streams
 - Understand the two types of streams (IO and file) and the operators and functions associated with them
 - The open() function. Understand what happens in cases where the file (either input or output) does as well as does not exist.
 - Extraction (>>) and Insertion (<<) operators - Understand the function and workings of each. Understand how the extraction operator deals with whitespace.

- The `get()` function - Understand when and how to use it. How is it different that the extraction operator?
- Numeric Manipulators - `setw()` and `setprecision()`. Understand the use of each. You should be able to write simple examples of each to print out columns of data with a given number of decimal places
- The `eof()` and `fail()` functions - What are the purposes of each and how do they differ of other, similar functions.
- The `peek()` and `putback()` functions - Understand their usage and how they relate to, and can be used with, the `get()` function.
- Reading until input file failure
 - Be prepared to write code to read in until input file failure for simple cases, using both extraction and the `get()` function. Understand why and how to do a priming read. What happens if we don't do a priming read? See course note slide 4.21 for a good "skeleton" of a while loop that reads until input file failure.

Chapter 5

- Booleans
 - Understand how the computer represents false and true (0 and every other number except 0)
 - Be able to write (and read) boolean expressions using the 6 C++ relational operators, `==`, `>`, `<`, `!=`, `>=`, and `<=`. Particularly, make sure you understand the difference between `=` and `==` (and what if you accidentally use `=` instead of `==`).
 - Be able to write (and read) boolean using the 3 boolean operators, `&&`, `||` and `!`.
 - Understand the idea of short circuit evaluation? When can it be applied (in other words, how can we take advantage of it) and what advantages does it have?
- Selection Statements
 - If's
 - Understand how the `if`, `if...else`, and `if...else if` statements work and what each's use is. Why and when do we use an `if...else if` instead of multiple `if`'s?
 - Be able to construct single `if` statements from nested `if`'s using boolean operators
 - Switch
 - Understand the form and function of the `switch` statement
 - Understand the advantages and, *particularly the disadvantages*, of the `switch` statement. Why can't we always use the `switch` statement?

Chapter 6

- While loops
 - Understand how to construct and use while and do while loops. This includes count and even controlled looping. What is the difference between a while and a do while loop?
- For loops
 - Understand how to construct and use for loops. When would you use a for loop over a while loop (and vice versa)?

Chapter 7

- Understand the advantages of a function. When do we use functions?
- Parts of a function
 - Understand the three parts associated with a function - the prototype, invocation, and declaration. What happens if one (namely, the prototype) is missing?
 - Be able to construct simple functions.
- Scoping
 - Understand the different types of scoping - global and local - and the rules that apply to each.
 - Understand name precedence
- Parameter passing methods
 - Understand the three different parameter passing methods - pass by value, pass by reference, and pass by constant reference. What are the differences between them? What are the advantages (and dangers!) of each?
 - Be able to apply the different methods to simple examples.

For this exam, reviewing the homework and the in-class review exercises is probably one of the primary (though not only) ways of studying. I'm not out to trick you on this! I recognize that this material can be very hard, especially when learned in such a short timeframe. You will be expected to apply your knowledge (particularly with arithmetic, loops, and functions), which may include writing short amounts of code. But you will not be asked to solve some complex problem that you have never seen before (I'll wait until the final for that! ;-)).