CS 215 Study Guide
Midterm Written Exam
Spring 2023

Exam format

- **Thursday, March 2, 2023**
- **75 minutes (normal Lecture time and Location)**
- 5 True or False questions (10%)
- 30 multiple-choice questions(60%)
- 3 short-answer questions (9%)
  - They require reading code and show the output.
- 3 coding questions (21%)
  - Define functions only.
  - Not a complete program.
  - Function header and implementation.
- 1 bonus coding question (extra 10%)

Topics

General topics it would be good to review for each chapter:

- Definitions and rules.
- Technical names of things.
- Syntax of C++ constructs.
- Meaning of C++ constructs (reading code).
- System classes, functions, and header files.
- Common errors.
- Algorithms.

Topics

- Computers, programming and programming languages (chapter 1)
- Variables and data types (chapter 2)
- Logical expressions and conditional statements (chapter 3)
- Loops (chapter 4)
- Functions (chapter 5)
- Arrays and Vectors (chapter 6)
- Streams and files (chapter 7)

Chapter 2

- Types
  - int, double (float), char, bool, string
  - Literals of each type.
  - When to use each type.
  - Differences between integers and floating point.
- Variables
  - Identifier, type, scope, address, value.
  - Declaration and definition.
  - Local and global variables.
- Arithmetic operations
  - Operations `+ - * / %`
  - Operator precedence.
  - Integer division—when does it happen?

Chapters 3 and 4

- Conditionals
  - `if`, `if-else`, nested `if`s, *switch*.
  - Body is one statement or a *{* block *}*
  - Boolean expressions: `&& || !`
  - Common errors: extra semicolon, missing curly braces.
  - Errors: tests that are always false or always true.
- Loops
  - For, while, and do-while loops.
  - Differences between types of loops.
  - Converting between for and while loops.
  - Scope of loop variables.
  - For loop bounds: how many iterations?

Chapter 5

- Functions
  - Syntax for function definitions.
  - Syntax for prototypes.
  - Parameters and arguments (what's the difference?)
  - Return values and types (including `void`).
  - The `return` statement.
  - Deciding what parameters and return type a function needs.
  - Call-by-value: copies the argument.
  - Call-by-reference: use the argument itself (why and how?)
  - `const` call by reference

Chapter 6

- Arrays
  - Syntax for arrays: `int mylist[12];`
  - Accessing: `a[i]`. Valid indexes *i* range from 0 to size - 1.
  - Looping over arrays: `for (int i = 0; i < size; i++)`
    - In reverse: `for (int i = size - 1; i >= 0; i--)`
  - Limitations: capacity must be known in advance.
  - Partially-filled arrays
    - Companion variable for the (current) size.
    - Capacity (maximum size) is still fixed.
    - Inserting and removing: shift the following elements.
  - Arrays as function arguments.
    - Passed as call-by-reference (even without &).
    - Also need to pass in the current size (possible capacity for some cases)
    - Cannot return an array.

Chapter 6

Vectors

- Syntax: `vector<int> mylist;`
- With an initial capacity: `vector<int> mylist(12);`
- Advantages compared to arrays.
  - Don't have to know the capacity: can shrink and grow.
  - Keeps track of its own length: no companion variable.
  - Can be returned from a function normally.
- Header file `<vector>`.
- Functions: `pop_back`, `push_back`, `size`.
  - Must use `push_back` to increase the size!
- Vectors as function arguments.
  - Compare pass-by-value and pass-by-reference
  - Does not need to pass in the current size (automatically tracked by size() function)
  - Can return a vector!

Chapter 7

- Streams
  - Header files `iostream`, `iomanip`, `ifstream`
  - Insertion operator `<<`
    - I/O manipulators: `setw`, `setprecision`, `endl`
  - Extraction operator `>>`
    - Skips initial whitespace.
    - Reads one item.
    - The rest is buffered, including the newline.
    - Sets a fail state on error.
  - The fail state: cannot use the stream until it is fixed.
    - Detecting: `fail` method.
    - Clearing errors: `clear` and `ignore`.
  - `getline` function: `getline(cin, stringvar)`
  - Reading in a loop and detecting errors.

Chapter 7

- File streams
  - Types `ifstream`, `ofstream`.
  - Opening and closing.
    - `ifstream infile; infile.open("outfile.txt");`
    - `infile.close();`
  - Reading from or writing to a file stream.
    - Exactly the same as using `cin` or `cout`.
    - Just use the name of the stream variable: `infile >> age`
  - What happens if a file is missing?
    - `ifstream`: A fail state (check with `.fail()`)
    - `ofstream`: No problem, the file is created.
  - Stream parameters: must be call-by-reference.
    - `istream` and `ostream` to accept `cin` and `cout` too.