

## CSCI 261 Exam 2 Review Topics

This is **not** a complete list, but these are the big, critical ideas you should know.

**C++ Strings:** `.length()`; `[]` array like access.

**Functions in C++:** What are function headers, definitions, and prototypes? How to write function prototypes; definition of `void` functions; how to write empty argument lists; using `return` in functions; how to call functions; scope of variables and parameters declared within functions; passing parameters by value and by reference; passing `const` references; how are arrays passed to functions; how can `const` be used with array parameters to functions; why is it wise to make the `size` or `count` parameter of an array accepting function a `const` parameter? What syntax is used to pass arrays as function parameters; providing 2d arrays as function parameters (all but first dimension size must be provided); function signatures; partially filled arrays and constructing functions for them; functions should not return arrays; overloaded functions; what is a header file; `static const` global variables in header files (useful for array sizes, project data); best practices for passing the three different data types (fundamental types, arrays, objects) to functions as parameters; using `srand()`, and `rand()`; the bubble sort algorithm; the binary search algorithm.

**C++ Classes:** How to declare a class; what are class constructors; how are constructors prototyped; what does `public` mean; what does `private` mean; what is a default constructor; how are default constructors used in `main`; how are user defined (parameterized) constructors used in `main`; what is a calling object; what does it mean when a function is written in class scope; what are the three different ways `const` may be used in the header of functions written in class scope; what is the difference between a class and an object instance or variable of the class; what is a “helper” function; what are accessor functions; how are “setters” prototyped and written; how are “getters” prototyped and written; what are the three steps to writing a solid input function (read to local variables checking for input failure, validate new data, store local values into object); declaring and accessing `static const` class data of fundamental **integral** variable types; in what ways should input and output functions be similar? The three steps for correctly reading object data from an input method<sup>1</sup>; using `infile.setstate( ios::failbit );` in input methods on failure; how are array data members declared and initialized in classes; how are “setters” and “getters” different for `private` array data members; how are array of classes declared and initialized; how are member functions (or data members) accessed for elements of object arrays.

---

<sup>1</sup>First, read data into **local** variables; second, check that local variables represent a valid state for the object; third, store the local variable values into the appropriate data members (or set `ios::failbit` otherwise).