

CSCI 261 Final Review Topics

The Final Exam is Cumulative!

Random Numbers in C++: What C++ library (`#include <.>`) contains the C++ random number functions; what function is used to specify the random number seed; what type of argument(s) does this function accept; what is the return type of this function; what function provides random number to an application; what type of argument(s) does this function accept; what is the return type of this function; if two identical seeds are provided to two independent runs of a C++ application, what is expected of the results.

Operator Overloading: What are the binary arithmetic and relational operators; how are operators prototyped; what types of operators have conventional or standardized return types; overloaded operators written by the non-class author; the `friend` keyword; overloaded operators written by the class author; overloading `operator<<` and `operator>>`; the three steps for correctly reading object data in `operator<<`¹; using `infile.setstate(ios::failbit);` in `operator<<` on input failure; overloading arithmetic and relational operators for a class.

Pointers: How to declare pointers and pointers with `const`; how to initialize pointer values; how to use `&` in code; pointer arithmetic (how do pointer values increment when added with 1); how pointers may be used as 1d arrays; how `*` dereferences pointers..

Dynamic Memory in C++: How to declare pointers and initialize their value in one C++ statement; allocating memory for one object: `int* p(new(nothrow) int(0));`; allocating memory for an array of objects: `int* q(new(nothrow) int[SIZE]);`; checking pointer values against `NULL` and `exit(1)` to handle errors. `delete p;` and `delete[] q;` (what happens when memory is deleted, when to use the two different versions of `delete`).

¹First, 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).