Programming Concepts & Data Structure from the Bottom-up

Part 1: LC-3

- Assembly language programming & process
- Memory-mapped I/O: input from keyboard, output to monitor
- TRAPs & Subroutines, Interrupts & Exceptions
- Stacks

Part 2: C

- Built-in data types, operators, scope
- Functions & run-time stack
- Pointers & arrays
- Recursion: searching, sorting, backtracking
- I/O: streams and buffers, read from / write to file
- User-defined data types: enum, struct, union
- Dynamic memory allocation
- Linked data structures: linked list (stack, queue) & trees

Part 3: C++

- Class (encapsulation, inheritance, abstraction)
- Pass by value /(const) reference / address
- Virtual function, operator overload, template (polymorphism)

Part 1: LC-3 Review

- Address space: **2^{16} locations**, addressability: **16 bits**
- General-purpose registers: **R0, R1, … R7**
- Special-purpose register: **PC, IR**
- Input from keyboard: **KBDR/KBSR**
- Output to monitor: **DDR/DSR**
- Operate instructions: **ADD, AND, NOT**
- Data movement instructions: **LD, LDI, LDR, LEA, ST, STR, STI**
- Control instructions: **BR, JSR/JSRR, JMP, RET, TRAP, RTI**
- Condition codes: **N** (negative), **Z** (zero), **P** (positive)
- TRAPs: In, GETC, OUT, PUTS (uses **R0**; **R7** is modified after call)
- Subroutines: callee-save vs. caller-save, nested subroutine needs to save R7
- Interrupts: external event, supervisor vs. user stack, RTI instruction
- Exceptions: internal event for handling errors
- Stack: FILO, overflow, underflow, **R6 – stack ptr, R5 – frame ptr**

Part 2: C Review

- Scope: **local** vs. **global** variables (determined by location of declaration)
- Storage class: **static** (retains value, global data area) vs. **automatic** (stack)
- Control structures: conditionals (**if, if-else, switch**); loops (**for, while, do-while**)
- Functions & run-time stack (**C to LC-3**)
- Pointer: **address** of a variable in memory
- Array: **a list** of values arranged **sequentially** in memory
- Pass by **value** vs. pass by **reference** (pointer)
- Pointer Array **Duality** (int array[10] = {1,2,3,4,5,6,7,8,9}; int *ptr = array;)
- Recursion: **base case**(s) and **recursive case**(s)
- File I/O: fopen, fclose, fscanf, fprintf
- Linked lists & trees (pointer, struct, dynamic memory allocation)

Part 3: C++ Review

- Class vs. Struct: **4 features of OOP**
- Dynamic memory allocation/deallocation: **new**, **delete**
- Basic I/O: **std**, **cin**, **cout**
- Pass by value vs. pass by address vs. pass by (const) reference
- Operator and function overloading
- Base class & derived Class: access identifier (**public**, **protected**, **private**)
- Virtual function and virtual function table: **static vs. dynamic binding**
- Function and class templates: **separate type with container**
- Big three: copy ctor (**deep vs. shallow copy**), dtor, copy assignment operator
- Implicit '**this**' pointer: a pointer to the invoking object
- **Vectors**: dynamic arrays, elements are stored in consecutive locations
- **Lists**: doubly linked lists, elements are allocated individually
- **Iterators**: mechanism used to minimize an algorithm's dependency on data structure on which it operates