**C++**

* Structs & classes are interchangeable (for the most part) in C++
* Use increment operators as a standalone statement, not part of a long expression
* Use for loops when possible
* #include <string> instead of #instead <string.h>
  + Imports C++ String class, not C string library
* Write at beginning using namespace std
* In C++ strings can be compared with == (unlike in Java)
* **Namespaces:**

namespace distance {

int length = 1;

}

int main() {

using namespace distance;

//length is now a keyword

}

* **Input/Output**
* NL – new line char
* WS – white space (space, tab, NL, etc.)
* EOF – end of file
* cin – from istream class; cout – from ostream class
* cout << “m” + “g” << endl; ← illegal because string literals are treated as char\*, and char\* cannot be added
  + Instead do cout << (string)”m” + (string)”g” << endl;
* Input: cin >> var1 >> var2 – reads next 2 tokens
* Use cin.eof() or cin.fail() to check for end of file
  + Eof() and fail() don’t return true until you’ve attempted to read beyond the end of file
  + Call eof() after an attempted read but before using the value read
  + cin >> next returns cin.fail() when coerced into a bool – use this to check EOF
* getline(cin, line) – read each character (including non-NL WS chars)
* cin >> line – move input stream marker past all WS cars to the next non-WS char
  + Then read each non-WS char into line
* **File I/O**
* #include <fstream>
* Create ifstream & ofstream objects:

ifstream inputFile (filename.c\_str()); //argument needs to be char\*

ifstream >> var; //etc.

ifstream.close();

* F
* **Arrays/Vectors**
* C-style arrays – a[i] = a + i (pointer)
  + Does not check if memory at a[i] actuallys belongs to a
  + Must state max size (“extent”) – must be a runtime constant
* C++ arrays – vectors
  + Manipulate dynamically

vector<int> vInts;

vInts.size();

vInts.at(index);

* + Or manipulate statically

Vector<string> months (12);

Months[0] = “Jan”;

* **Stack**
* Push(e) – add element e on top of all other elements
* Pop() – return & remove the top element
  + Can only get the value of top element & not remove it, or only remove the element w/o getting its value
* Size always <= capacity
  + First active element is v(0); last active element is v(v.size() – 1)
* When capacity is full and more elements are added (size > capacity), vector is reassigned with double the previous capacity
* For vector as a stack, use push\_back(e), pop\_back(), back() to add/remove/get the element at the end of the list
  + Push & pop only change size by 1, capacity may double
* Adding a new element is occasionally a linear-time operation instead of constant time
  + “Amortized constant time”