# Data Structures with C++: CS189

Lecture 3-1: STL and Functors

## Standard Template Library (STL)

- An "Abstract Data Type" (ADT) is not data on its own
  - It is a... shape or use for data
- STL is a set of ADT's that have been developed over the years
  - First a bunch of programmers noticed they were all doing the same tasks with data all the time
  - Second they gave each of these data patterns names so they were easier to talk about
- We will look inside a few of these because this is school, but in real life you'll just use STL every day

#### Vector

vector<int> myVector;

- An array that automatically resizes if you put too much data in it
  - You will never make a vanilla array again in your life
- Many of the commands for vector are shared across many ADTs
  - begin, end, size, front, back, push\_back, pop\_back, clear
- And some are more vector specific
  - o at, [], resize, reserve, insert

## Templates

vector<int> myVector;

- A vector is "abstract" because it doesn't mean anything.
  - O I have two vectors?
- An ADT is commonly read with the phrase "of what" so it has data
  - vector of ints
- We want vectors to hold anything
- The <> there can be thought of as a "wildcard" to declare what type is inside
  - Once set, can't be changed on this vector

# How Templates Work

- A templated class is written in a single h file
- Since ADT is abstract, it doesn't exist so it doesn't compile
- When the compiler sees you declaring a "vector of ints", it copypastes the vector file in to a new file
  - Then it does a find-replace using what you have in <>
- If you had ten different vector-ofs, there would be ten (hidden) classes
  - vector\_int, vector\_float, vector\_Student

#### List

list<int> myList;

- A long chain of data without special powers
- Again has many methods shared across
   STL
  - begin, end, front, back, size, push\_back, pop\_back, clear
- Has a few different ones
  - o push\_front, pop\_front
- And is missing some from vector
  - o at, []

#### Vector vs List

- One of the key topics in this class is deciding which ADT is right for the situation
  - Vector and List share so many commands, how do decide?
- We need to talk about Big O (how we measure speed) and see inside some of STL before we can really know
  - With information hiding, we may never
- For now, just say that vector is faster at accessing data, and list is better at adding and removing

# Stack and Queue

queue<int> myQueue;

stack<int> myStack;

- Some ADTs are more specialized but limited to make them easy to use
- A Stack is like a stack of paper. You put paper on top and can only see that one
  - o top, push, pop
- A Queue is like a bank. The first one put in is the first to come out
  - front, back, push, pop
- You have no idea if there is a Vector or a List inside doing the work
  - Information Hiding!

#### Set

set<int> mySet;

- A Set just holds a bunch of data like List, but it is faster at searching
  - o insert, erase, clear, find
  - Slower at adding
  - Data is sorted without extra work
- Can only handle duplicate values
   (dupes) if you use MultiSet, which is
   slower
  - Continuing theme of how you need to know why you are using an ADT in order to pick the best one

## Map

map< string, int > myMap;

- For keeping track of data pairs
  - Every student's grade
  - Bank Account's balance
  - Family member's age
- Sometimes called Associative Array, since it associates a value of the first template type with a value of the second
- Reintroduces some commands from Vector
  - insert, erase, find, at, []

## Using Map

```
// Tracking ages
```

```
map<string, int> myMap;
myMap["Bob"] = 34;
myMap["Alice"] = 24;
```

```
cout << myMap["Bob"];</pre>
```

- Worth taking a moment on the syntax since Map is possibly the most useful
- A normal array uses numbers as indexes
   0, 1, 2...
- A Map lets you put whatever you want in the brackets. Just needs to match the type
- Inside the brackets is the "key", and it tracks the "value"
  - "Key value pair" is another name for Map

## **Unordered Map**

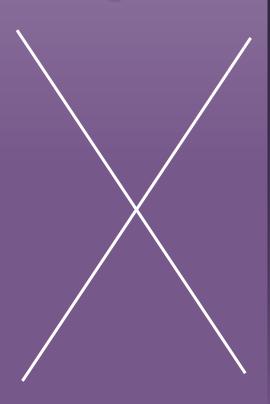
unordered\_map <string, int> myMap;

- This is a Map where the data is not in order
  - Same commands
  - Set and Map sort things automatically to make searching faster
- Unordered Map can't do searches, but it gains instant lookups
  - Again, you need to think about why you are using it before you can pick the right ADT for your code

# Proceed to Canvas Quiz

You don't need to understand how these work in order to use them.

## Algorithms



- The shape and purpose of ADTs are well established, so it makes sense there would be a set of common operations
- The <algorithms> include is separate from the include for the actual container
  - <vector>, <list>, etc
- You will use them in real life, but because the entire point of the class is to learn how things work, using them is forbidden
  - Asking you to write a sorting algorithm and having you just call sort() would be pointless

# End

If the quiz went well you can do 80% of the HW now.