Standford Library to Standard Library Cheat sheet
COMP220 Data Structures

Fall 2016

This document list standard library equivalents to the collections classes in the Stanford C++ library used in the text book. Handling of special cases and errors may differ between the Stanford and Standard equivalents presented here. You should verify the specifications with the appropriate resource.

Vector

The C++ std::vector class[1] can be used in place of the Stanford C++ Vector class[2].

[1] `http://www.cplusplus.com/reference/vector/vector/`

[2] pg 197

Figure 1: Vector to std::vector

Operation	Stanford	Std
Empty Constructor	Vector<t>()	std::vector<t>()
Fill Constructor	Vector<t>(size,val)	std::vector<t>(size,val)
Size Query Method	v.size()	v.size()
Empty Predicate	v.isEmpty()	v.empty()
Element Selector	v.get(idx)	v.at(idx)
	v[idx]	v[idx]
Element Mutator	v.set(idx,val)	v[idx] = val
Add to Back	v.add(val)	v.push_back(val)
Insert at	v.insertAt(idx,val)	v.insert(std::begin(v) + idx, val)
Remove at	v.removeAt(idx)	v.erase(std::begin(v) + idx)
Erase all	v.clear()	v.clear()
Concatenate	v + w	v.insert(std::end(v),std::begin(w),std::end(w))
Repeated Add	v += a,b,c,d,e ...	N.A.

Stack

The std::stack class[3] can be used in place of the Stanford Stack class[4].

[3] `http://www.cplusplus.com/reference/stack/stack/`

[4] pg 211

Figure 2: Stack to std::stack

Operation	Stanford	Std
Empty Constructor	Stack<t>()	std::stack<t>()
Size Query	s.size()	s.size()
Empty Predicate	s.isEmpty()	s.empty()
Push Element	s.push(val)	s.push(val)
Pop Element	s.pop()	s.pop()
Peek at Top	s.peek()	s.top()
Remove all Elements	s.clear()	N.A.

Queue

The std::queue class[5] can be used in place of the Stanford C++ Queue class[6].

[5] http://www.cplusplus.com/reference/queue/queue/

[6] pg 217

Operation	Stanford	Std
Empty Constructor	Queue<t>()	std::queue<t>()
Size Query	q.size()	q.size()
Empty Predicate	q.isEmpty()	q.empty()
Enqueue Element	q.enqueue(val)	s.push_back(val)
Dequeue Element	s.dequeue()	s.pop_front()
Peek at Front	s.peek()	s.front()
Remove all Elements	s.clear()	N.A.

Figure 3: Queue to std::queue

Map

The std::map class[7] can be used in place of the Stanford C++ Map class[8].

[7] http://www.cplusplus.com/reference/map/map/

[8] pg 226

Operation	Stanford	Std
Empty Constructor	Map<kt,vt>()	std::map<kt,mt>()
Size Query	m.size()	m.size()
Empty Predicate	m.isEmpty()	m.empty()
Select Element	q.get(key)	m.at(key)
	m[key]	m[key]
Add Element	m.put(key,val)	m.insert(std::pair<kt,vt>(key,val))
	m[key] = val	m[key] = val
Remove Key+Value	m.remove(key)	m.erase(key)
Key Containment Predicate	m.containsKey(key)	m.count(key)
Remove all Elements	m.clear()	m.clear()

Figure 4: Map to std::map

Alternatively, one can use std::unordered_map[9] to improve the performance of certain map operations.

[9] http://www.cplusplus.com/reference/unordered_map/unordered_map/

Set

The std::set class[10] can be used in place of the Stanford C++ Set class[11]. The std::includes, std::set_intersection, std::set_difference, and std::set_union functions are found in the *algorithm* library[12]

[10] http://www.cplusplus.com/reference/set/set/

[11] pg 232

[12] http://www.cplusplus.com/reference/algorithm/

Figure 5: Set to std::set

Operation	Stanford	Std
Empty Constructor	Set<t>()	std::set<t>()
Size Query	s.size()	s.size()
Empty Predicate	s.isEmpty()	s.empty()
Add Element	s.add(val)	s.insert(val)
Remove Element	s.remove(key)	s.erase(key)
Containment Predicate	s.contains(val)	s.count(key)
Remove all Elements	s.clear()	s.clear()
Is Subset of	s.isSubsetOf(r)	std::includes(std::begin(s),std::end(s), std::begin(r),std::begin(s))
Get First	s.first()	*(std::begin(s))
Union	s + r	std::union(std::begin(s),std::end(s), std::begin(r),std::end(r),std::begin(result_set))
Intersection	s * r	std::set_intersection(std::begin(s),std::end(s), std::begin(r),std::end(r),std::begin(result_set))
Difference	s − r	std::set_difference(std::begin(s),std::end(s), std::begin(r),std::end(r),std::begin(result_set))

Alternatively, one can use std::unordered_set[13] to improve the performance of certain set operations.

[13] http://www.cplusplus.com/reference/unordered_set/unordered_set/