

# \*\*\* CS 106X MIDTERM REFERENCE SHEET \*\*\*

You can perform a for-each loop over any collection other than Stack and Queue. `for (type name : collection) { ... }`

## Vector<T> Members ("vector.h") (5.1)

<code>v.add(value);</code> or <code>v += value;</code>	appends to end of vector	O(1)
<code>v.clear();</code>	removes all elements	O(1)
<code>v.get(index)</code> or <code>v[index]</code>	returns value at given index	O(1)
<code>v.insert(index, value);</code>	inserts at given index, shifting subsequent values right	O(N)
<code>v.isEmpty();</code>	returns <code>true</code> if there are no elements	O(1)
<code>v.remove(index);</code>	removes value at given index, shifting subsequent values left	O(N)
<code>v.set(index, value);</code> or <code>v[index] = value;</code>	replaces value at given index	O(1)
<code>v.size();</code>	returns number of elements	O(1)
<code>v.toString();</code>	returns string representation of elements such as "{1, 2, 3}"	O(N)

## Grid<T> Members ("grid.h") (5.1)

<code>g.fill(value);</code>	set every cell to store a given value	O(R*C)
<code>g.get(row, col)</code> or <code>g[row, col]</code>	returns value stored at given row/column	O(1)
<code>g.inBounds(row, col)</code>	returns <code>true</code> if given row/column index is within (0, 0) ... (R, C)	O(1)
<code>g.numCols()</code> // or <code>g.width();</code>	returns number of columns C	O(1)
<code>g.numRows()</code> // or <code>g.height();</code>	returns number of rows R	O(1)
<code>g.resize(nCols, nRows);</code>	changes grid to have the given number of rows/columns; wipes all data	O(R*C)
<code>g.set(row, col, value);</code> or <code>g[row][col] = value;</code>	changes value stored at given row/column	O(1)

## Stack<T> Members ("stack.h") (5.2)

<code>s.clear();</code>	removes all elements
<code>s.push(value);</code>	adds given value on top of the stack
<code>s.pop();</code>	remove/return top value from stack; pop/peek throw exception if empty
<code>s.peek();</code>	return top value without removing
<code>s.isEmpty();</code>	returns <code>true</code> if there are no elements
<code>s.size();</code>	returns number of elements
<code>s.toString();</code>	string (right=top) such as "{1, 2, 3}"

## Queue<T> Members ("queue.h") (5.3)

<code>q.clear();</code>	removes all elements	O(N)
<code>q.enqueue(value);</code>	adds value to back of queue	O(1)
<code>q.dequeue();</code>	remove/return value from front; dequeue/peek throw if empty	O(1)
<code>q.peek();</code>	return front without removing	O(1)
<code>q.isEmpty();</code>	returns <code>true</code> if no elements	O(1)
<code>q.size();</code>	returns number of elements	O(1)
<code>q.toString();</code>	(left=front) e.g. "{1, 2, 3}"	O(N)

## Set<T> and HashSet<T> Members ("set.h", "hashset.h") (5.5)

<code>s.add(value);</code> or <code>s += value;</code>	adds to set; if a duplicate, no effect	set O(log N), hash O(1)
<code>s.clear();</code>	removes all elements	O(N)
<code>s.contains(value)</code>	returns <code>true</code> if value is found in the set	set O(log N), hash O(1)
<code>s.isEmpty();</code>	returns <code>true</code> if there are no elements	O(1)
<code>s.isSubsetOf(s2)</code>	returns <code>true</code> if <code>s2</code> contains all elements of <code>s</code>	O(N)
<code>s.remove(value);</code> or <code>s -= value;</code>	removes value from set, if present	set O(log N), hash O(1)
<code>s.size();</code>	returns number of elements	O(1)
<code>s.toString();</code>	returns string such as "{1, 2, 3}"	O(N)
<code>s1 == s2, s1 != s2</code>	operators for set equality testing	O(N)
<code>s1 + s2, s1 += s2;</code>	operators for union; adds elements of <code>s2</code> to <code>s1</code>	O(N)
<code>s1 * s2, s1 *= s2;</code>	intersection; removes all from <code>s1</code> not found in <code>s2</code>	O(N)
<code>s1 - s2, s1 -= s2;</code>	difference; removes all from <code>s1</code> that are found in <code>s2</code>	O(N)

## Lexicon Members ("lexicon.h") (5.5)

<code>l.add(word);</code>	adds a word; if a duplicate, no effect	O(log N)
<code>l.clear();</code>	removes all words	O(N)
<code>l.contains(word)</code>	returns <code>true</code> if the word is found in the lexicon	O(log N)
<code>l.containsPrefix(text)</code>	returns <code>true</code> if any word starts with this prefix text	O(log N)
<code>l.isEmpty();</code>	returns <code>true</code> if there are no words in the lexicon	O(1)
<code>l.remove(word);</code>	removes word from lexicon, if present	O(log N)
<code>l.size();</code>	returns number of words	O(1)
<code>s.toString();</code>	returns string such as "{a, ball, cat, zebra}"	O(N log N)

# \*\*\* CS 106X MIDTERM REFERENCE SHEET \*\*\*

## Map<K, V> and HashMap<K, V> Members ("map.h", "hashmap.h") (5.4)

<code>m.clear();</code>	removes all key/value pairs	O(N)
<code>m.containsKey(key)</code>	returns <b>true</b> if map contains a pair for the given key	map O(log N), hash O(1)
<code>m.get(key)</code> or <code>m[key]</code>	returns value paired with the given key (a default value if the key is not present)	map O(log N), hash O(1)
<code>m.isEmpty()</code>	returns <b>true</b> if there are no key/value pairs	O(1)
<code>m.keys()</code>	returns a <b>Vector</b> copy of all keys in the map	O(N)
<code>m.put(key, value)</code> or <code>m[key] = value;</code>	adds a pairing of the given key to the given value	map O(log N), hash O(1)
<code>m.remove(key);</code>	removes any existing pairing for the given key	map O(log N), hash O(1)
<code>m.size()</code>	returns number of key/value pairs	O(1)
<code>m.toString()</code>	returns string representation such as "{a:90, d:60, c:70}"	O(N)
<code>m.values()</code>	returns a <b>Vector</b> copy of all values in the map	O(N)

A for-each loop on a map iterates over the *keys*, not the *values*.

## String Members and Utility Functions (<string>, "strlib.h") (3.2)

<code>str.at(index)</code> or <code>s[index]</code>	character at a given 0-based index in the string
<code>str.append(str);</code>	add text to the end of a string ( <i>in-place</i> )
<code>str.c_str()</code>	returns the equivalent C string
<code>str.compare(str)</code>	return -1, 0, or 1 depending on relative ordering
<code>str.erase(index, length);</code>	delete text from a string starting at given index ( <i>in-place</i> )
<code>str.find(str)</code> <code>str.rfind(str)</code>	returns the first or last index where the start of the given string or character appears in this string ( <b>string::npos</b> if not found)
<code>str.insert(index, str);</code>	add text into a string at a given index ( <i>in-place</i> )
<code>str.length()</code> or <code>str.size()</code>	number of characters in this string
<code>str.replace(index, len, str);</code>	replaces <b>len</b> chars at given index with new text ( <i>in-place</i> )
<code>str.substr(start, length)</code> or <code>str.substr(start)</code>	returns the next <b>length</b> characters beginning at index <b>start</b> (inclusive); if <b>length</b> is omitted, grabs from <b>start</b> to the end of the string
<code>endsWith(str, suffix)</code> <code>startsWith(str, prefix)</code>	returns <b>true</b> if the string begins or ends with the given prefix/suffix
<code>integerToString(int)</code> , <code>stringToInteger(str)</code> <code>realToString(double)</code> , <code>stringToReal(str)</code>	returns a conversion between numbers and strings
<code>equalsIgnoreCase(str1, str2)</code>	<b>true</b> if <b>s1</b> and <b>s2</b> have same chars, ignoring casing
<code>stringSplit(str, separator)</code>	breaks apart a string into a vector of smaller strings based on a separator
<code>toLowerCase(str)</code> , <code>toUpperCase(str)</code>	returns an upper/lowercase version of a string
<code>trim(str)</code>	returns string with any surrounding whitespace removed

## char Utility Functions (<cctype>) (3.3)

<code>isalpha(c)</code> , <code>isdigit(c)</code> , <code>isspace(c)</code> , <code>isupper(c)</code> , <code>ispunct(c)</code> , <code>islower(c)</code>	returns <b>true</b> if the given character is an alphabetic character from a-z or A-Z, a digit from 0-9, an alphanumeric character (a-z, A-Z, or 0-9), an uppercase letter (A-Z), a space character (space, \t, \n, etc.), respectively
<code>tolower(c)</code> , <code>toupper(c)</code>	returns lower/uppercase equivalent of a character

## istream Members (<iostream>) (Ch. 4)

<code>f.fail()</code>	returns <b>true</b> if the last read call failed (e.g. EOF)
<code>f.open(filename);</code>	opens file represented by given string
<code>f.close();</code>	stops reading file
<code>f.get()</code>	reads and returns 1 character
<code>getline(f&amp;, str&amp;)</code>	reads line of input into a string by reference; returns a <b>true/false</b> indicator of success
<code>f &gt;&gt; variable</code>	reads a whitespace-separated token of data from input into a variable

## Random Numbers ("random.h")

<code>randomBool()</code>	returns a random <b>bool</b> of <b>true/false</b> with 50/50% probability
<code>randomChance(probability)</code>	returns a random <b>bool</b> of <b>true/false</b> with the given probability of <b>true</b> from 0..1
<code>randomInteger(min, max)</code>	returns a random integer in the range [ <i>min-max</i> ], inclusive
<code>randomReal(Low, high)</code>	returns a random real number in the range [ <i>low-high</i> ), up to but not including <i>high</i>