

*** CS 106B/X MIDTERM REFERENCE SHEET ***

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

* All Big-Oh runtimes listed are average-case; some methods perform differently under various cases.

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

<code>v.add(val);</code> or <code>v += val;</code>	appends to end of vector	$O(1)$ *
<code>v.clear();</code>	removes all elements	$O(1)$
<code>v.get(i)</code> or <code>v[i]</code>	returns value at given index	$O(1)$
<code>v.insert(i, val);</code>	inserts at given index, shifting subsequent values right	$O(N)$
<code>v.isEmpty()</code>	returns true if there are no elements	$O(1)$
<code>v.remove(i);</code>	removes value at given index, shifting subsequent values left	$O(N)$
<code>v.set(i, val);</code> or <code>v[i] = val;</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(val);</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 true 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, val);</code> or <code>g[row][col] = val;</code>	changes value stored at given row/column	$O(1)$

Stack<T> and Queue<T> Members ("stack.h") (5.2, 5.3)

<code>s.clear();</code>	removes all elements	<code>q.clear();</code>	removes all elements	$O(N)$
<code>s.push(val);</code>	adds given value on top of the stack	<code>q.enqueue(val);</code>	adds value to back of queue	$O(1)$
<code>s.pop()</code>	remove/return top value from stack; pop/peek throw exception if empty	<code>q.dequeue()</code>	remove/return value from front; dequeue/peek throw if empty	$O(1)$
<code>s.peek()</code>	return top value without removing	<code>q.peek()</code>	return front without removing	$O(1)$
<code>s.isEmpty()</code>	returns true if	<code>q.isEmpty()</code>	returns true if no	$O(1)$

	there are no elements		elements
<code>s.size()</code>	returns number of elements	<code>q.size()</code>	returns number of elements $O(1)$
<code>s.toString()</code>	string (right=top) such as "{1, 2, 3}"	<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(val);</code> or <code>s += val;</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(val)</code>	returns true if value is found in the set	set $O(\log N)$, hash $O(1)$
<code>s.first()</code>	returns first element from set (does not remove it)	set $O(\log N)$, hash $O(1)$
<code>s.isEmpty()</code>	returns true if there are no elements	$O(1)$
<code>s.isSubsetOf(s2)</code>	returns true if <code>s2</code> contains all elements of <code>s</code>	$O(N)$
<code>s.remove(val);</code> or <code>s -= val;</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 true if the word is found in the lexicon	$O(\log N)$
<code>l.containsPrefix(text)</code>	returns true if any word starts with this prefix text	$O(\log N)$
<code>l.isEmpty()</code>	returns true 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 106B/X 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 true 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 (or a default value such as 0, false, "" if key is not present)	map $O(\log N)$, hash $O(1)$
<code>m.isEmpty()</code>	returns true if there are no key/value pairs	$O(1)$
<code>m.keys()</code>	returns a Vector copy of all keys in the map	$O(N)$
<code>m.put(key, val);</code> or <code>m[key] = val;</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 Vector 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(i)</code> or <code>s[i]</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(i, 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 (or string::npos if not found)
<code>str.insert(i, 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(i, len, str);</code>	replaces <i>len</i> 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 <i>length</i> characters beginning at index <i>start</i> (inclusive); if <i>length</i> is omitted, grabs from <i>start</i> to the end of the string
<code>endsWith(str, suffix)</code> <code>startsWith(str, prefix)</code>	returns true if the string begins or ends with the given prefix/suffix
<code>integerToString(int), stringToInteger(str)</code>	returns a conversion between numbers and strings

`realToString(double), stringToReal(str)`
`equalsIgnoreCase(str1, str2)`
`stringSplit(str, separator)`

`toLowerCase(str), toUpperCase(str)`
`trim(str)`

true if *s1* and *s2* have same chars, ignoring casing
 breaks apart a string into a vector of smaller strings
 based on a separator

 returns an upper/lowercase version of a string
 returns string with any surrounding whitespace
 removed

char Utility Functions (<cctype>) (3.3)

`isalpha(c), isdigit(c), isspace(c),`
`isupper(c), ispunct(c), islower(c)`

`tolower(c), toupper(c)`

returns true 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

 returns lower/uppercase equivalent of a character

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

`f.fail()`

`f.open(filename);`
`f.close();`
`f.get()`
`getline(f&, str&)`

`f >> variable`

returns true if the last read or open call failed (e.g.
 EOF, or file-not-found)
 opens file represented by given string
 stops reading file
 reads and returns 1 character
 reads line of input into a string by reference;
 returns a true/false indicator of success

 reads a whitespace-separated token of data from input
 into a variable

Random Numbers ("random.h")

`randomBool()`

`randomChance(probability)`

`randomInteger(min, max)`

`randomReal(low, high)`

returns a random bool of true/false with 50/50%
 probability
 returns a random bool of true/false with the given
 probability of true from 0..1
 returns a random integer in the range [*min-max*],
 inclusive
 returns a random real number in the range [*low-high*],
 up to but not including *high*

File Utility Functions ("filelib.h")

`fileExists(name)`

`getHead(name), getTail(name)`

`isDirectory(name)`
`isFile(name)`
`listDirectory(name)`

whether this file exists on the disk
 separate a file path into the directory and file part; for
 "a/b/c/d.txt", head is "a/b/c", tail is "d.txt"
 returns whether this file name represents a directory
 returns whether this file name represents a regular file
 returns a Vector with the tails of all the file names
 contained in the given directory