Cheat-sheet

C++

Vectors (std::vector)

- `push_back(element)`: Adds an element to the end.
- `pop_back()`: Removes the last element.
- `at(index)`: Accesses element at index with bounds checking.
- `size()`: Returns the number of elements.
- `empty()`: Checks if the vector is empty.
- `clear()`: Removes all elements.
- `insert(position, element)`: Inserts element at specified position.
- `erase(position)`: Removes element at specified position.
- `begin()`, `end()`: Iterators for the beginning and end of vector.
- `sort(begin, end, comparator)`: Sorts elements using a custom comparator.
- `reverse(begin, end)`: Reverses the order of elements.

Strings (std::string)

- `size()`: Returns the length of the string.
- `empty()`: Checks if the string is empty.
- `clear()`: Clears the string.
- `substr(start, length)`: Extracts a substring.
- `find(substring)`: Returns the index of the first occurrence of substring.
- `replace(start, length, new_string)`: Replaces part of the string.
- `append(string)`: Appends another string.
- `compare(string)`: Compares two strings.

Priority Queues (std::priority_queue)

- `push(element)`: Adds an element to the queue.
- `pop()`: Removes the highest priority element.
- `top()`: Returns the highest priority element.
- `size()`: Returns the number of elements.
- `empty()`: Checks if the queue is empty.
- `emplace(args...)`: Constructs element in-place.

Maps (std::map)

- `insert({key, value})`: Inserts a key-value pair.
- `erase(key)`: Removes the element with specified key.
- `find(key)`: Returns an iterator to the element with key.
- `at(key)`: Accesses element with key, with bounds checking.
- `size()`: Returns the number of elements.
- `empty()`: Checks if the map is empty.
- `clear()`: Removes all elements.
- `count(key)`: Returns the number of elements with key.

Sets (std::set)

- `insert(element)`: Inserts an element into the set.
- `erase(element)`: Removes the element from the set.
- `find(element)`: Returns an iterator to the element if found.
- `count(element)`: Returns 1 if element is present, 0 otherwise.
- `size()`: Returns the number of elements.
- `empty()`: Checks if the set is empty.
- `clear()`: Removes all elements.

Sorting Priority Queues and Maps with Custom Comparator

To sort a priority queue or map with a custom comparator, you can define your comparator function and pass it as the third argument to `std::priority_queue` or `std::sort` for maps.

Example of sorting a priority queue with a custom comparator:

```C++
struct CustomComparator {
    bool operator()(const T& a, const T& b) const {
        // Define your comparison logic here
    }
};

std::priority_queue<T, std::vector<T>, CustomComparator> pq;
```

Example of sorting a map with a custom comparator:

```cpp
struct CustomComparator {
    bool operator()(const Key& a, const Key& b) const {
        // Define your comparison logic here
    }
};

std::map<Key, Value, CustomComparator> mp;
```

Java

Arrays

- `Arrays.sort(array)`: Sorts the array in ascending order.
- `Arrays.sort(array, comparator)`: Sorts the array using a custom comparator.
- `Arrays.toString(array)`: Converts the array to a string representation.

ArrayList (java.util.ArrayList)

- `add(element)`: Adds an element to the end of the list.
- `remove(index)`: Removes the element at the specified index.
- `get(index)`: Returns the element at the specified index.
- `size()`: Returns the number of elements in the list.
- `isEmpty()`: Checks if the list is empty.
- `clear()`: Removes all elements from the list.

LinkedList (java.util.LinkedList)

- `add(element)`: Adds an element to the end of the list.
- `addFirst(element)`, `addLast(element)`: Adds element to the beginning or end.
- `removeFirst()`, `removeLast()`: Removes and returns the first or last element.
- `getFirst()`, `getLast()`: Returns the first or last element without removing it.
- `size()`, `isEmpty()`, `clear()`: Same as ArrayList.

PriorityQueue (java.util.PriorityQueue)

- `offer(element)`: Adds an element to the queue.
- `poll()`: Removes and returns the highest priority element.
- `peek()`: Returns the highest priority element without removing it.

- `size()`, `isEmpty()`, `clear()`: Same as ArrayList.

HashMap (java.util.HashMap)

- `put(key, value)`: Associates the specified value with the specified key.
- `get(key)`: Returns the value associated with the specified key.
- `containsKey(key)`, `containsValue(value)`: Checks if the map contains the key or value.
- `remove(key)`: Removes the mapping for the specified key.
- `size()`, `isEmpty()`, `clear()`: Same as ArrayList.

HashSet (java.util.HashSet)

- `add(element)`: Adds an element to the set.
- `contains(element)`: Checks if the set contains the element.
- `remove(element)`: Removes the element from the set.
- `size()`, `isEmpty()`, `clear()`: Same as ArrayList.

Sorting with Custom Comparator

- For sorting arrays or lists with a custom comparator, use `Arrays.sort(array, comparator)` or `Collections.sort(list, comparator)` respectively, where `comparator` is an instance of `Comparator<T>` or `Comparable<T>`.

Here's an example of sorting an array with a custom comparator:

```java
Integer[] array = {3, 1, 4, 1, 5, 9};
Arrays.sort(array, (a, b) -> Integer.compare(b, a));  // Sort in descending order
System.out.println(Arrays.toString(array));  // Output: [9, 5, 4, 3, 1, 1]
```

This cheat sheet covers common methods and operations for arrays, lists, queues, maps, and sets in Java, including sorting with custom comparators.