# STL Containers and Operations Cheat Sheet

## Vector

- Declaration: `std::vector<int> vec = {1, 2, 3, 4, 5};`  
- Add element: `vec.push\_back(6);`  
- Remove last element: `vec.pop\_back();`  
- Access element: `vec[0];`  
- Sort: `std::sort(vec.begin(), vec.end());`  
- Sum: `std::accumulate(vec.begin(), vec.end(), 0);`  
- Find: `std::find(vec.begin(), vec.end(), 3);`  
- Copy: `std::copy(vec.begin(), vec.end(), std::back\_inserter(vecCopy));`

## List

- Declaration: `std::list<int> lst = {1, 2, 3, 4, 5};`  
- Add element at end: `lst.push\_back(6);`  
- Add element at beginning: `lst.push\_front(0);`  
- Remove last element: `lst.pop\_back();`  
- Remove first element: `lst.pop\_front();`

## Deque

- Declaration: `std::deque<int> deq = {1, 2, 3, 4, 5};`  
- Add element at end: `deq.push\_back(6);`  
- Add element at beginning: `deq.push\_front(0);`  
- Remove last element: `deq.pop\_back();`  
- Remove first element: `deq.pop\_front();`

## Queue

- Declaration: `std::queue<int> que;`  
- Add element: `que.push(1);`  
- Access first element: `que.front();`  
- Remove first element: `que.pop();`

## Stack

- Declaration: `std::stack<int> stk;`  
- Add element: `stk.push(1);`  
- Access top element: `stk.top();`  
- Remove top element: `stk.pop();`

## Set

- Declaration: `std::set<int> st = {5, 3, 1, 4, 2};`  
- Add element: `st.insert(6);`  
- Remove element: `st.erase(3);`

## Unordered Set

- Declaration: `std::unordered\_set<int> ust = {5, 3, 1, 4, 2};`  
- Add element: `ust.insert(6);`  
- Remove element: `ust.erase(3);`

## Map

- Declaration: `std::map<int, std::string> mp = {{1, "one"}, {2, "two"}, {3, "three"}};`  
- Add element: `mp[4] = "four";`  
- Remove element: `mp.erase(2);`  
- Access element: `mp[1];`

## Unordered Map

- Declaration: `std::unordered\_map<int, std::string> ump = {{1, "one"}, {2, "two"}, {3, "three"}};`  
- Add element: `ump[4] = "four";`  
- Remove element: `ump.erase(2);`  
- Access element: `ump[1];`

## Array

- Declaration: `std::array<int, 5> arr = {1, 2, 3, 4, 5};`  
- Access element: `arr[0];`

## Tuple

- Declaration: `std::tuple<int, double, std::string> tpl = std::make\_tuple(1, 3.14, "hello");`  
- Access elements: `std::get<0>(tpl);`, `std::get<1>(tpl);`, `std::get<2>(tpl);`

## Algorithms

### Sort  
- Function: `std::sort(container.begin(), container.end());`  
- Example: `std::sort(vec.begin(), vec.end());`  
  
### Accumulate  
- Function: `std::accumulate(container.begin(), container.end(), initial\_value);`  
- Example: `int sum = std::accumulate(vec.begin(), vec.end(), 0);`  
  
### Find  
- Function: `std::find(container.begin(), container.end(), value);`  
- Example: `auto it = std::find(vec.begin(), vec.end(), 3);`  
  
### Copy  
- Function: `std::copy(source\_container.begin(), source\_container.end(), std::back\_inserter(destination\_container));`  
- Example:   
 ```cpp  
 std::vector<int> vecCopy;  
 std::copy(vec.begin(), vec.end(), std::back\_inserter(vecCopy));  
 ```  
  
### Fill  
- Function: `std::fill(container.begin(), container.end(), value);`  
- Example:   
 ```cpp  
 std::fill(vec.begin(), vec.end(), 0);  
 ```  
  
### Remove  
- Function: `std::remove(container.begin(), container.end(), value);`  
- Example:  
 ```cpp  
 vec.erase(std::remove(vec.begin(), vec.end(), 3), vec.end());  
 ```  
  
### Reverse  
- Function: `std::reverse(container.begin(), container.end());`  
- Example:  
 ```cpp  
 std::reverse(vec.begin(), vec.end());  
 ```  
  
### Transform  
- Function: `std::transform(container1.begin(), container1.end(), container2.begin(), unary\_op);`  
- Example:  
 ```cpp  
 std::transform(vec.begin(), vec.end(), vec.begin(), [](int x) { return x \* 2; });  
 ```