```
Auto [a,b] = pair // a,b are newly created variables. Can only unpack fixed len at compile time(no vector unpack)
swap(a,b) // a,b = b,a swap(v1,v2)// swap vectors
v1=move(v2); v2 is moved to v1 by reference

/***** vector *****/
vector<int> v={1,2,3};
vector<int> v(n, 10);  vector<vector<int>> v(N, vector<int>(M,0)); // N*M vector
v.push_back(value); v.pop_back(); // push_back is by value, not reference
v.insert(v.begin() + index, value); // insert in the middle
v1.insert(v1.end(), v2.begin(), v2.end());
sort(v.begin(), v.end(), greater<int>()); // largest first
sort(v.begin(), v.end(), [](int a, int b){return a>b); // largest first with lambda
Auto maxval = *max_element(v.begin(), v.end()); // get max/min
Auto sum = reduce(v.begin(), v.end());
Bool exist = find(v.begin(), v.end(), val)!=v.end();
reverse(v.begin(), v.end()); //reverse
/***** deque *****/
deque<int> d; // (queue with two ends)
d.push_front(value); d.pop_front();
d.push_back(value);  d.pop_back();
Auto e = d[3] // access ith element
/***** stack *****/
stack<int> s;
int res = s.top(); s.pop(); //pop doesn't return
/***** priority_queue *****/
priority_queue<int> q; // max heap by default, max at front
priority_queue<int, greater<int>> // min heap
priority_queue<pair<int,int>, greater(pair<int, int>)> // min heap with pair
q.push(val);
int res = q.top(); q.pop();// pop doesn't return

/***** pair, tuple *****/
pr=make_pair(2,3);
p.first, p.second;
tp=make_tuple(1,4); get<0> tp;

/***** hashmap *****/
// dictionary
unordered_map<int, int> s; s[1]=2; s.erase(1);
// set
unordered_set<int> s; s.insert(1); s.erase(1);
Bool exist = s.count(val)>0 // check val in hashmap
set1.merge(set2); //set1 Union set2, goes into set1
set_intersect(s1.begin(), s1.end(), s2.begin(), s2.end());
/***** bisect *****/
Auto lower = lower_bound(v.begin(), v.end(),value); // return iterator at left most insert position
Auto upper = upper_bound(v.begin(), v.end(),value); // return iterator at right most insert position

/***** bst *****/
map<string, string> m; //BST with key and value
m["key"]=value; or  m.insert(    pair<string, string>("key", "value")    )
m.erase("key")
Bool exists=  (m.find("key") !=m.end());
set<string, string> // BST with only value

/***** string *****/
string s1 = "Geek" + "s";
to_string(num); //int to string. For char, char(32) <-> int('4')
stoi(s); // string to int
string(1,ch); // char to int
string r = s1.substr(startPos, length);
```