vector

```
// initialize array
int arrary[length];
int arrary[] = {1, 2, 3, 4};
int arrary[length] = {1, 2, 3, 4};
int array[lengthA][lengthB];  // only declare

// initialization
vector<int> v;
vector<int> v = {0,1,2,3,4,5};
vector<int> v(7);  // 7 elements are 0 by default
vector<int> v(7, 1); // 7 elements are one

// resize
v.resize(n);   // only change the capacity
v.resize(n, val);   // change the capacity and initialize new elements

// access and add element
v.front();  // access the first element
v.back();   // access the last element
v.push_back();
v.pop_back();   // remove the last element

// insert
v.insert(it_pos, ele);
v.insert(it_pos, count, ele);
v.insert(it_pos, eles.begin(), eles.end());

// sort
sort(v.begin(), v.end());
// count element frequency
count(v.begin(), v.end(), element);

// relative distance
int idx = distance(v.begin(), v.begin() + 5);

/* math */
// min value of a vector
iterator min_pos = min_element(v.begin(), v.end());
int min_val = *min_pos;
// sum value of a vector
int sum = accumulate(v.begin(), v.end(), sum0);
```

unordered_set

```
s.insert(element);  // insert the element
s.count(element);  // count the freq of the element
// erase the elements
s.erase(s.begin());   // erasing by iterator
s.erase("France");    // erasing by key
s.erase(s.find("Japan"), s.end() ); // erasing by range
```

stack

```
st.push(element);  // insert the element
st.top();  // access the top element
st.pop();  // remove the top element
```

queue

```
q.push(element);  // insert element
q.emplce(element);  // construct and insert element
q.front();  // access next element
q.back();   // access last element
q.pop();    // remove next element
q1.swap(q2);    // exchange the contents of the queues
```

unordered_map

```
// delete by key
u.erase(key)
```

string

```
// find
// return the position of 1st char of the first match; otherwise returns string::npos
s.find(substring);
s.find(substring, start_index);
s.find(substring, start_index, len); // length of subtring to match
// substring
s.substr(start_index, length=optional);

// add char(s)
s.push_back(char);
s += char;
s.append(string);
s.append(times, char);

// conversion
stoi(str);  // string to int
to_string(num); // any type of values to string
```

priority_queue

```
// declare
priority_queue<int> pq; // largetest by default
priority_queue<int, vector<int>, greater<int> > pq; // smallest at top

pq.push(val);
pq.top(); // access the top element
pq.pop(); // remove the top element
```