STL 简介

vector

```
vector, 变长数组，倍增的思想
    size() 返回元素个数
    empty() 返回是否为空
    clear() 清空
    front() / back()
    push_back() / pop_back()
    befin() / end()
    []
    支持比较运算，按字典序
```

pair

```
pair<int, int>
    first 第一个元素
    second 第二个元素
    支持比较运算，以first为第一关键字，以second为第二关键字（字典序）
```

string

```
string, 字符串
    size() / length() 返回字符串的长度
    empty()
    clear()
    += string
    substr(起始下标, (子串长度)) 返回子串
    c_str() 返回字符串所在字符数组的起始地址
```

priority_queue

```
priority_queue, 优先队列, 默认是大顶堆
    priority_queue<int> heap;
    size()
    empty()
    push() 插入一个元素
    top() 返回堆顶元素
    pop() 弹出堆顶元素
    定义成小顶堆的方式：priority_queue<int, vector<int>, greater<int>> q;
```

自定义比较方式：

```
struct Node {
    int value;
    int priority;
};
```

```
// 自定义比较方式（小顶堆，优先级低的先出）
struct Compare {
    bool oprator() (const Node& a, const Node& b) {
        return a.priority > b.priority;
    }
}

priority_queue<Node, vector<Node>, Compare> p;

// 值得注意的是，sort自定义比较方式与此是不同的：
sort(array, array+6, Compare());
```

stack

```
stack, 栈
    size()
    empty()
    push()
    top()
    pop()
```

queue

```
queue, 队列
    size()
    empty()
    push() 向队尾插入一个元素
    front() 返回队头元素
    back() 返回队尾元素
    pop() 弹出队头元素
```

deque

```
deque, 双端队列
    size()
    empty()
    clear()
    front() / back()
    push_back() / pop_back()
    push_front() / op_front()
    begin() / end()
    []
```

set, multiset

```
set, map 基于平衡二叉树（红黑树），动态维护有序序列
    size()
    empty()
    clear()
    begin() / end()  ++, --  返回前驱和后继，时间复杂度 O(logn)
```

```
    insert() 插入一个数
    find() 查找一个数
    count() 返回某一个数的个数
    erase()
        (1) 输入是一个数x，删除所有x O(k + logn)
        (2) 输入是一个迭代器，删除这个迭代器
    lower_bound() / upper_bound()
        lower_bound(x) 返回大于等于x的最小的数的迭代器
        upper_bound(x) 返回大于x的最小的数的迭代器
```

map / multimap

```
map, multimap 基于平衡二叉树（）
    size()
    empty()
    clear()
    ++. -- 返回前驱和后继，时间复杂度 O(logn)

    insert() 插入一个pair
    erase() 输入pair或者迭代器
    find()
    [] 时间复杂度是O(logn)，注意multimap不支持此操作
    lower_bound() / upper_bound()
```

unordered_...

```
unordered_set, unordered_map, unordered_multiset, unordered_multimap 哈希
    和上面类似，增删改查的时间复杂度是O(1)
    不支持 lower_bound() / upper_bound()，迭代器的++/--
```

bitset

```
bitset 压位
    bitset<10000> s;
    ~, &, |, ^
    >>, <<
    ==, !=
    []

    count() 返回有多少个1

    any() 判断是否至少有一个1
    none() 判断是否全为0

    set() 把所有位置成1
    set(k, v) 把第k位置成v
    reset() 把所有位置成0
    flip() 等价于~
    flip(k) 把第k位取反
```