

# C++ with STL templates

---

[compare](#)

[sort](#)

[vector](#)

[链表](#)

[stack](#)

[deque](#)

[multiset](#)

[priority\\_queue](#)

[set & unordered\\_set](#)

[map & unordered\\_map](#)

[cmath](#)

## compare

▼ Comparison Functions

C++ |

```
1  #include<algorithm>
2
3  int greater=max(a,b);
4  int smaller=min(a,b);
```

## sort

```
1  #include<algorithm>
2  vector<int> arr;
3  arr.push_back(2);
4  arr.push_back(4);
5  arr.push_back(3);
6  arr.push_back(7);
7  sort(arr.begin(),arr.end());//默认从小到大排序
8  vector<vector<int>> grid;
9  grid.push_back([3,2]);
10 grid.push_back([5,3]);
11 grid.push_back([1,3]);
12 sort(grid.begin(),grid.end());//二维数组按照数组元素的第一个数字从小到大排序
13 //自定义排序：（优先队列的自定义排序也可以这么写）
14 struct cmp{
15     bool operator()(vector<int>& a, vector<int>&b){//这里的两个参数随机应变
16         return a[1]<b[1];//按照a、b数组的第二个元素从小到大排序
17         //a<b: 从小到大;a>b: 从大到小
18     }
19 }
20 sort(grid.begin(),grid.end(),cmp);
```

## vector

```
1  #include<vector>
2
3  vector<int> v;//声明
4  vector<int> v(20);//声明, 且里面开始即留20个空间,默认初始化为0 (及可以直接用v[i]访问)
5  vector<int> v(20,5);//默认初始化为20个5
6  //声明二维数组:
7  vector<vector<int>>(n,vector<int>(m,2));//n*m大小二维数组全部填满2
8  vector<vector<int>> move={{0,1},{1,0},{0,-1},{-1,0}};//在岛屿类题目中常见
9  //
10 v.push_back(num);//向容器末尾插入元素num
11 v.pop_back();//清除容器末尾的元素
12 v.back();//返回容器v最后一个元素
13 v.resize(n)//将v内部的元素个数改为n个, 多出的删去, 少的补(默认补0), 本来就有且不出超出的部分保留
14 v.resize(n,num)//补的部分补numv, 其余同上
15 v.insert(v.begin(),num);//将num插入头部
16 v.insert(v.begin()+n,num);//将num插入某位置
17 v.empty();//是空则返回true
18 v.clear();//清空所有值
19 v.size();//v中的元素个数
20 v.capacity();//v占据的内存(注意和size()的区别)
21 //声明二维数组:
22 vector<vector<int>> v(n,vector<int>(m));
23 //在类的private中声明vector类型并初始化:
24 vector<int> v=vector<int>(5,5);
25 //迭代器的使用:
26 vector<int>::iterator it=v.begin();
27 it++;//迭代器向后移动一位
28 it<v.end();//判断结束
29 *it=10;//赋值给相对应迭代器的位置
30 //返回空数组:
31 return vector<int>(0);
```

## 链表

官方给的默认链表格式, 自己也要会写

▼ LinkedList

C++ |

```
1 struct ListNode{
2     int val;
3     ListNode* next;
4     ListNode(): val(0), next(nullptr){}
5     ListNode(int v): val(v), next(nullptr){}
6     ListNode(int v, ListNode* nex):val(v), next(nex){}
7 };
```

## stack

▼ stack

C++ |

```
1 stack<int> stk;
2 stk.push(1);
3 stk.empty();
4 stk.top();
5 stk.pop();
```

## deque

▼ deque

C++ |

```
1 #include<deque>
2
3 queue<int>q;
4
5 // 常用函数: front指队头, back指队尾 (新插入的)
6 q.push_back(num);
7 q.push_front(num);
8 q.pop_back();
9 q.pop_front();
10 int num=q.front();
11 int num=q.back();
12 int n=q.size();
13 bool flag=q.empty();
```

## multiset

## multiset

C++

```

1  #include<set>
2
3  multiset<int> mset;
4  mset.insert(3); //插入元素:0(log(n))
5  mset.insert(5);
6  mset.insert(1);
7  mset.insert(2);
8  mset.erase(2); //删除元素:0(log(n)), 注意会删除所有等于改值的
9  mset.erase(mset.begin()+1); //删除某迭代器位置上的元素, 即使有重复也只会删除一个
10 *mset.begin(); //有序集合的首位元素 (利用迭代器)
11 *mset.rbegin(); //有序集合的末尾元素 (利用迭代器)
12 //自定义排序: 类似sort, 不再重述
13 mset.find(3); //找某元素, 找不到返回mset.end()
14 mset.count(3); //某元素数量
15 mset.upper_bound(5); //第一个大于5的元素的迭代器
16 mset.lower_bound(5); //第一个大于等于5的元素的迭代器
17 mset.count(5); //5的个数

```

## priority\_queue

### priority\_queue

C++

```

1  # include<queue>
2  priority_queue<int> q; //默认大根堆
3  q.push(); //插入元素
4  q.pop(); //弹出元素
5  q.top(); //返回堆顶元素
6  q.empty(); //是否为空
7  q.size(); //返回元素个数
8
9  //内部元素优先级设置:
10 priority_queue<int> q; //默认大根
11 priority_queue<int, vector<int>, less<int> > q; //大根
12 priority_queue<double, vector<double>, greater<double>> q; //小根

```

## set & unordered\_set

▼ set

C++ |

```
1 // 集合
2 set<int> s;
3
4 // 常用函数
5 int n=s.size();
6 s.insert(num);
7 s.erase(num);
8 s.count(num); // 常用来判断有无这一元素
9
10 // 遍历元素
11 set<int>::iterator it;
12 for(it = s.begin(); it!=s.end(); it++){ //自动排序元素
13     cout<<*it<<endl; // 这里的it是个指向数据的指针
14 }
15 //c++ 11
16 for(auto it : s){
17     cout<<it<<endl; // 这里的it就是数据本身
18 }
```

▼ unordered\_set

C++ |

```
1 unordered_set<int> uset;
2 uset.insert(x);
3 uset.erase(x); //删去元素
4 uset.find(x); //找到返回迭代器, 否则返回uset.end()
5 uset.count(x); //返回x的数目, 0或者1
6 uset.size(); //返回元素个数
7 uset.clear(); //清空
8 uset.empty(); //空返回1
9 //遍历:
10 unordered_set<int> set = {9,5,9,8,1,2,3,5,6,1,2,3,4,5,6,7,4,3,3};
11 for(unordered_set<int>::iterator it = set.begin(); it!=set.end(); it++)
12 {
13     cout<<*it<<" ";
14 }
```

## map & unordered\_map

## ▼ unordered\_map

C++

```
1 unordered_map<int, string> umap;//声明
2 //注意：直接对未初始化的umap[i]++, 直接生成默认0再操作
3 umap.emplace(key,value);//向umap中添加键值对
4 umap[1]="aaa";//向umap中添加键值对
5 umap.find(key);//找到此key并返回其迭代器, 否则返回umap.end()
6 //(注意是找到key不是找到其对应的value)
7 umap.count(key);//找到key返回1, 否则0
8 umap.size();//返回哈希表大小
9 //遍历:
10 for(pair<int,string>key:umap)
11 {
12     function(key.first,key.second);
13 }
14 //OR:
15 for(unordered_map::iterator it = map.begin();it!=map.end();it++){
16     int front = it->first;//key
17     int end = it->second;//value
18 }// (用迭代器进行遍历)
```

## ▼ map

C++

```
1 // map
2 map<string, int> m;
3
4 // 常用函数
5 int n=m.size();
6 m[key]=value; // 最简单的插入方法 (个人认为)
7 m.erase(key);
8 s.count(key); // 常用来判断有无这一元素
9
10 // 遍历元素
11 map<string, int>::iterator it;
12 for (it = m.begin(); it != m.end(); it++) {
13     cout<<it->first<<" "<<it->second<<endl;
14 }
15 // c++ 11
16 for(auto it : m){
17     cout<<it.first<<" "<<it.second<<endl;
18 }
```

## cmath

```
▼ cmath C++ |

1  #include<cmath>
2
3  int abs(int i); //返回整型参数i的绝对值
4  double fabs(double x); //返回双精度参数x的绝对值
5  long labs(long n); //返回长整型参数n的绝对值
6
7
8  double exp(double x); //返回指数函数e^x的值
9  double log(double x); //返回logex的值
10 double log10(double x) //返回log10x的值
11 double pow(double x, double y) //返回x^y的值
12 double pow10(int p) //返回10^p的值
13
14
15 double sqrt(double x) //返回+√x的值
16
17
18 double acos(double x) //返回x的反余弦arccos(x)值,x为弧度
19 double asin(double x) //返回x的反正弦arcsin(x)值,x为弧度
20 double atan(double x) //返回x的反正切arctan(x)值,x为弧度
21 double cos(double x) //返回x的余弦cos(x)值,x为弧度
22 double sin(double x) //返回x的正弦sin(x)值,x为弧度
23 double tan(double x) //返回x的正切tan(x)值,x为弧度
24
25
26 double hypot(double x, double y) //返回直角三角形斜边的长度(z),x和y为直角边的长
    度,  $z^2 = x^2 + y^2$ 
27
28
29 double ceil(double x) //返回不小于x的最小整数
30 double floor(double x) //返回不大于x的最大整数
31
32
33 int rand() //产生一个随机数并返回这个数
34
35 srand(time(NULL)); //srand 设置随机种子
36 double atof(char *nptr) //将字符串nptr转换成浮点数并返回这个浮点数
37 double atol(char *nptr) //将字符串nptr转换成长整数并返回这个整数
38 double atof(char *nptr) //将字符串nptr转换成双精度数,并返回这个数,错误返回0
39 int atoi(char *nptr) //将字符串nptr转换成整型数,并返回这个数,错误返回0
```