C++好用的函数

upper/lower_bound

upper_bound（第一个大于/小于x）, lower_bound（第一个大于等于/小于等于x）

以lower_bound举例：

```
//  a为vector
int idx = lower_bound(a.begin(), a.end(), x) - a.begin();
//  a为数组
int idx = lower_bound(a, a + n, x) - a;
//  第一个小于等于x的
int idx = lower_bound(a.begin(), a.end(), x, greater<int>())
```

min max

```
min({a, b, c})
```

max/min_element

```
cout << *max_element(a, a + n);
```

prev/next_permutation

```
do{

} while(next_permutation(a+1,a+n+1));
```

> Tips: next_permutation实际上获取的上当前顺序的next排列，如果要获得全排列，要先确保数组有序

to_string&stoi&atoi

```
string pi = to_string(3.1415926); 数字转字符串
int a = stoi(str);  字符串转int
int a = atoi(str.c_str());  字符串转char*转int
```

__gcd __lg

__gcd返回最大公因数，__lg返回当前数二进制下有几位，即最高位是第几位（从第0位开始）

```
__lg(12) = 3; // 12 = (1100)
```

reverse

写字符串常用

```
reverse(s.begin(), s.end());
```

__builtin_popcount

统计有多少个位为1

iota

对[a, a + n]进行递增赋值

```
iota(a + 1, a + n + 1, 10) // 10, 11, 12, ......
```

STL

string

```
string substr(int pos, int n = npos)  pos开始的n个字符组成的子串
int find(string s, int pos = 0)从pos开始查找s在当前字符串中的位置，失败返回string::npos
char *c_str() 返回c字符串的指针，内容与string相同
s += c; string后加char/string
```

bitset

```
bitset<N> 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() 所有位置成1
set(k, v) 第k位置成v
reset() 所有位置成0
flip() 取反
flip(k) 第k位
```

priority_queue

```
push() 插入
top() 堆顶
pop() 弹出堆顶
小根堆：priority_queue<int, vector<int>, greater<int> > q;
```

INT_MAX INT_MIN

库中自带的宏定义