头文件

```
#include<bits/stdc++.h>

#include<algorithm>
#include<unordered_map>
#include<map>
#include<string>
#include<iostream>
#include<queue>
#include<stack>
```

优先队列

```
#include<queue>
priority_queue<int> que;
//默认情况
//是大根堆 堆顶是最大的，所以叫做 **优先**队列


priority_queue<int, vector<int>, less<int>> que;//大顶堆   这样理解，从前到后 越来越小
priority_queue<int, vector<int>, greater<int>> que;//小顶堆    从前到后 越来越大
//第二个参数 是设置底层放置数据的容器

//自定义类型必须重载操作符
struct node {
    int x,y;

    bool operator< (const node &b) const {
        return x < b.x;
    } //构建大顶堆

    bool operator< (const node &b) const {
        return x > b.x;
    }//构建小顶堆
}
//修改重载 < 小于号的 比较规则
```

vector

初始化

```
vector<int> vec; //size = 0 cap = 0

vector<int> vec = {1,2,3,4,5,6};

vector<int> vec(nums.begin()+2, nums.end()-1);

vector<int> vec(7,0); //初始化为 包含7 个值为3 的int

vector<vector<int>> vec(m, vector<int>(n,0));
//m * n 的二维数组
```

pair

pair.first

pair.second

{xx,xx}

map遍历

```
 map<string, int>::iterator it;
for (it = m2.begin(); it != m2.end(); it++) {
    string s = it->first;
    printf("%s %d\n", s.data(), it->second);
}
```

读取一行数据

getline 可读取整行，包括前导和嵌入的空格，并将其存储在字符串对象中

```
#include<string>
getline(cin, inputLine);
```

getchar 读取一个字符

```
char test1 = getchar();
```

读取所有 整数数据

```
while(cin>>x)w[n++]=x;
    cnt=w[n-1];
    n--;
```

读取一整行 整数

```
getline (istream& is, string& str, char delim);
//读取一行直到 delim(不保存)  或者读取到行尾
```

```
#include<iostream>

int num;
vector<int> vec;
while(cin>>num) {
    vec.push_back(num);
    char ch = getchar();
    if(ch == '\n')
        break;
}
```

自定义排序

```
struct node {
    int index = 0;
    int cnt = 0;
};

bool cmp(node a, node b) {
    if (a.cnt == b.cnt) return a.index < b.index;
    else return a.cnt > b.cnt;
}
```

素数

```
bool isprime(int num){ //判断一个数是否是素数
    for(int i = 2; i * i <= num; i++){ //遍历到根号num
        if(num % i == 0) //检查有无余数
            return false;
    }
    return true;
}
```

小数位数控制

```
//cout
double x = 3.1415926
cout << setw(10) << x << endl;//控制输出的宽度，就是printf("%10d");
cout << setprecision(5) << x << endl;//限制输出有效数字的个数是5
cout << fixed << setprecision(3) << x << endl;//fixed表示控制小数点后面的数字，两个连用就是小数点后面保留三位小数
//输出结果分别是    3.14159  3.1416  3.142

//printf
printf("%3.0f",floatNum) //不保留小数
// %3.0f表明待打印的浮点数至少占3个字符宽，且不带小数点和小数部分，整数部分至少占3个位宽

printf("%6.2f".floatNum)//保留两位小数
// %6.2f 表明待打印的数至少占6个字符宽度（包括两位小数和一个小数点），且小数点后面有2位小数，小数点占一位，所以整数部分至少占3位
```

大小写转换

```
#include<ctype.h>

toupper();//如果 c 有相对应的大写字母，则该函数返回 c 的大写字母，否则 c 保持不变

tolower();
```

结构体构造函数

```
struct node {
    int data;
    string str;
    char x;
    node():data(),str(),x(){}
    node(int a, string b, char c):data(a),str(b),x(c){}
};
```

快排

```
//l 左边界 r 右边界 都能取到
void quickSort(vector<int>& nums, int l, int r) {
    if(l >= r) return;
    int i=l, j = r;
    swap(nums[(i+j)/2], nums[l]);
    while(i < j) {
        while(i < j && nums[j] >= nums[l]) j--;
        while(i < j && nums[i] <= nums[l]) i++;
        swap(nums[i], nums[j]);
    }
```

```
    swap(nums[i], nums[l]);
    quicksort(nums, l, i-1);
    quicksort(nums, i+1, r);
}
```

```
void quicksort(vector<int>& nums, int l, int r) {
    if(l < r){
        int pos = partition(nums,l,r);
        quicksort(nums,l,pos-1);
        quicksort(nums,pos+1,r);
    }
}

int partition(vector<int>& nums, int l, int r) {
    int i=l, j=r;
    swap(nums[(i+j)/2], nums[l]);
    while(i < j) {
        while(i < j && nums[j] >= nums[l]) j--;
        while(i < j && nums[i] <= nums[l]) i++;
        swap(nums[i], nums[j]);
    }
    swap(nums[i], nums[l]);
    return i;
}
```