结构体

各种函数（尤其是字符处理相关）

find() 查找字符串，找不到返回-1

sort()自定义cmp函数

getline(cin,s) 取一行，记得getchar()取回车

str.replace(pos,len,str)

替换，用str替换指定字符串从起始位置pos开始长度为len的字符

str=str.replace(str.find("a"),2,"#"); //从第一个a位置开始的两个字符替换成#

str=str.replace(str.begin(),str.begin()+5,"#"); //用#替换从begin位置开始的5个字符

substr()

截取字符串

c=s.substr(a,b)//截取s的下标a开始的b的字符（不足或b不写则到结尾）并赋值给c

算法

dfs（真的要命啊）、最短路、并查集、全排列全排列next\_permutation()

数学思想

\_\_gcd()辗转相除（最大公因数、最小公倍数）、差分、快速幂

有序数组或容器！！！

(lower\_bound(a,a+n,int num)-a):在数组a中找到第一个大于等于num的数字并返回其下标

int a[] = {1, 2, 3, 4, 5, 7, 8, 9};

printf("%d", lower\_bound(a, a + 8, 6) - a);//-a不能少，因为是地址相减

输出5，a->a+8中第一个大于等于6的位置是5

(upper\_bound(a,a+n,int num)-a):在数组a中找到第一个大于num的数字并返回其下标

substr(a,b):返回下标从a到b的字符串

string s = "asdfg";

cout << s.substr(1,2) << endl;//此时会输出sd

bool compare(int a,int b){

return a>b;//降序

}

sort(num,num+4,compare);

max\_element()，min\_element()：

这两个函数用来快速得到数组中的最大/小值和最大/小值下标

min\_element()和max\_element()的应用方法相同

int num[6]={1,0,5,2,5,0};

cout << \*max\_element(num,num+6) << endl;

//则此时输出的是num数组的最大值5

cout << max\_element(num,num+6)-num << endl;

//则此时输出的是num数组的最大值的下标2

去重函数

unique(data,data+n):将date数组去重，n是数组长度

注意：unique只是能对相邻元素去重，

用这个函数之前还需要使用sort()进行排序才行，更重要的是所谓的去重并没有将重复的内容删去，而是把他们都排在了最后面

\_\_gcd:这个函数来求最大公约数，调用一个万能头文件

int gc = \_\_gcd(4,8);

如果用不了参考

int gcd(int a,int b){

if(b==0) return a;

else

return gcd(b,a%b);

}

最小公倍数

int lcm(int a,int b){

return a/\_\_gcd(a,b)\*b;

}

swap(a,b)//a和b进行交换

next\_permutation(): 这是在STL中的，注意要先将数组用sort排个升序

//int型数组：

sort(num,num+3);

int num[3] = {5,2,6};

sort(num,num+3);

//先使用sort排个序

do{

for(int i=0;i<3;i++)

cout << num[i] << " ";

cout << endl;

}while(next\_permutation(num,num+3));

//string型:

string s;

cin >> s ;

sort(s.begin(),s.end());

//先使用sort排个序

do{

cout << s << endl;

}while(next\_permutation(s.begin(),s.end()));

字符串对比

compare(), strcmp()

string s1="asd", s2="asd";

s1.compare(s2);

/\*相等返回0, 大于返回1, 小于返回-1\*/

char s1[10]="asd", s2[10]="zxc"

strcmp(s1, s2);

/\*相等返回0\*/

字符串翻转

reverse() 函数没有返回值

int num[3]={ 1, 2, 3 };

reverse( num, num+3 );//3是数组长度

vector<int> v = { 1, 2, 3 };

reverse(v.begin(),v.end());

string str="asdf";

reverse(str.begin(),str.end());

在数组中查找某个值

find 能查到就返回下标地址，找不到返回数组长度

i 的位置=find(开始位置，结束位置+1，i )

int num[] = {1, 2, 3};

int \*index;

index = find(num, num+3, 3);//index是查找到3的所在地址

cout << index-num << endl;//用3所在地址减去数组首元素地址就是3的下标

index = find(num, num+3, 5);//找不到返回数组长度所在地址

cout << index-num << endl;

/\*这里输出数组长度3\*/

字符串长度

string型：s.length(), s.size()

char型：strlen(s)

sizeof(str)/sizeof(str[0])，s.length()

strlen()函数求出的字符串长度为有效长度，既不包含字符串末尾结束符 ‘\0’；

sizeof()操作符求出的长度包含字符串末尾的结束符 ‘\0’；

length()函数求出的字符串长度不包含字符串末尾结束符’\0’;

当在函数内部使用sizeof()求解由函数的形参传入的字符数组的长度时，得到的结果为指针的长度，既对应变量的字节数，而不是字符串的长度，此处一定要小心。

取字符

getline ：可接收空格，适用string

string str;

getline(cin,str);

getchar() ：接受一个字符，可接收空格，回车，只适用于char，常用于吃回车

cin.get ：可接收空格，只适用于char，获取指定数量字符

char ch[20];

cin.get(ch, 11);//获取10个字符

char ch1;

cin.get(ch1);//获取1个字符,相当于ch1 = cin.get()

cin.getline() ：可接收空格，只适用于char,用法和cin.get()差不多，获取指定数量字符

char a[20];

cin.getline(a, 20);

char m[3][20];

for(int i=0;i<3;i++)

cin.getline(m[i],20);

gets() ： 接收一个字符串，可以接收空格，只适用于char

char ch[20];

gets(ch);

容器

vector、stack、queue、map、pair、heap

一些函数用法还没记全

迭代器还不太会用 auto、iterator（这玩意记不清）

vector//一种容器,类似动态数组，但它的size可以动态改变

vector在获取元素和对最后一个元素的操作效率上更高；但对于中间元素的操作，性能则相对较差。

vector<int>v

vector<int >b(a) //将a中的元素复制到b中

vector<int>a(100,6)//定义100个值为6的元素

vector<string>b(a.begin(),a.end()) //将动态数组a的元素值复制到b中

v.begin()//返回指向迭代器第一个元素的指针

v.end()//返回指向迭代器最后一个元素的指针

v.size()//返回当前vector使用数据量的大小

v.empty()//测试vector是否为空

v.front()//返回第一个元素的值

v.back()//返回最后一个元素的值

v.push\_back(x)//在容器最后一个位置插入元素x

v.pop\_back(x)//删除最后一个元素

insert erase swap

v.clear()//清空

Iterators迭代器

vector<int>::iterator iter1;

for (iter1 = v.begin(); iter1 != v.end(); ++iter1) {

cout << ' ' << \*iter1 << endl; //[0,1,2,3,4,5,6,7,8,9]

}

queue

queue<int> q：建立一个队列q，其内部元素类型是int。

q.push(a)：将元素a插入到队列q的末尾。

q.pop()：删除队列q的队首元素。

q.front()：查询q的队首元素。

q.back()：查询q的队尾元素。

q.size()：查询q的元素个数。

q.empty()：查询q是否为空。

set

set<int>a;

a.insert()的作用就是往集合中添加元素，它的参数也是只有一个，就是你想要添加的元素，无返回值。

a.count()返回值要么是1要么是0

是查找出现在集合中的元素的个数，因为集合的互异性嘛，所以它的返回值要么是1要么是0，一般也可以用它来判断某个元素是否在该集合中。

a.begin()和a.end()它们是有返回值的，返回值分别是第一个指向该元素的迭代器和指向最后一个元素迭代器的下一个位置

a.erase函数是用来删除指定的元素，参数只有一个，并且是你想要删除的元素，无返回值

a.size()的作用是得出集合中的元素个数。

a.empty()是判断该集合是否为空。

a.clear()是用来清空集合中的元素。

stack

stack<int> S; //定义一个存储整数类型的栈

top()：返回一个栈顶元素的引用，类型为 T&。如果栈为空，返回值未定义。

push(const T& obj)：可以将对象副本压入栈顶。这是通过调用底层容器的 push\_back() 函数完成的。

push(T&& obj)：以移动对象的方式将对象压入栈顶。这是通过调用底层容器的有右值引用参数的 push\_back() 函数完成的。

pop()：弹出栈顶元素。

size()：返回栈中元素的个数。

empty()：在栈中没有元素的情况下返回 true。

emplace()：用传入的参数调用构造函数，在栈顶生成对象。

pair（底层使用结构体实现，存储一对数据）

pair<T1, T2> p1;

p1.first = 1;

p1.second = 2.5;

cout<<p1.first<<' '<<p1.second<<endl;

map 映射

map<typename1, typename2> mp;typename1是键的类型，typename2是值的类型。

map中的键是唯一的

map<char, int> mp;

mp['c'] = 20;

cout << mp['c']; //答案输出20

map可以使用it->first来访问键，it->second来访问值

map会以键从小到大的顺序自动排序

find()

map<char, int>::iterator it = mp.find('b');

cout << it->first << " " << it->second;

erase()

mp.erase(it)，it为需要删除的元素的迭代器

mp.erase(key)，key为要删除的映射的键

mp.erase(first, last)，其中，first为需要删除的区间的起始迭代器，last为需要删除的区间末尾迭代器的下一个地址，即为删除左闭右开的区间[first, last)

size()用来获得map中映射的对数

clear()用来清空map中的所有元素

unordered\_map

unordered\_map<string, int> umap; //定义

heap 堆

完全二叉树，父节点均不小于/大于子节点

make\_heap(v.begin(), v.end(), greater<int>());//生成一个小堆，第三个不加就默认最小堆，加greater<int>()则为最大堆

pop\_heap(v.begin(), v.end(), greater<int>()); // 将堆的第零个元素与最后一个元素交换

v.pop\_back(); // 删除最后位置的元素

list 序列

使用双链表实现