小技巧：内容太多可以使用Ctrl + F 直接搜索 加快查询速度(4.6更新)

✨1、vector 【可变数组】

数组加强版

①头文件：

#include<vector>

1

②初始化：

这个初始化比较详细，后面一些容器用法类似

#include<iostream>

#include<vector>

using namespace std;

int main () {

//几种初始化的方法

vector<int> a;//定义一个vector 未初始化 输出》 0

vector<int> a(3);//定义一个长度为3的vector 未初始化 输出》0 0 0

vector<int> a(10, 3); //定义一个长度为10，且每个数赋值为3

//将向量b中从下标0 1 2（共三个）的元素赋值给a，a的类型为int型

//它的初始化不和数组一样

vector<int>a(b.begin(),b.begin+3);

//从数组中获得初值

int b[7]={1,2,3,4,5,6,7};

vector<int> a(b,b+7）;

for(auto x : a) {//遍历输出

cout << x << " ";

}

return 0;

}

③size()

a.size( )//返回元素个数

1

④a.resize( )

a.resize( )//改变大小

1

⑤empty()

a.empty();

//判断a是否为空，空则返回true，非空则返回false

1

2

⑥front()和 back()

a.front();

//返回a的第1个元素,当且仅当a存在

a.back();

//返回vector的最后一个数

⑦倍增的思想

[C++]系统为某一程序分配空间的所需时间，与空间大小无关，与申请次数有关如申请一个空间为1000 和 空间为1 申请1000次的所需时间差别是很大的，申请次数越多，越耗时间

⑧clear()

a.clear();

//清空a中的元素

1

2

⑨支持比较运算

比较操作==，！=，<，<，<=，>，>=

int main () {

//支持比较运算

vector<int> a(4, 3), b(3, 4);

//a: 3 3 3 3 b:4 4 4

//比较原理字典序 (根据最前面那个判断，如果一样就往后比较)

if (a < b) {

puts("a < b");

}

return 0;

}

⑩push\_back()和pop\_back();

a.pop\_back();

//删除a向量的最后一个元素

a.push\_back(5);

//在a的最后一个向量后插入一个元素，其值为5

1

2

3

4

⑪begin()和end()

a.begin();// vector的第0个数

a.end();// vector的最后一个的数的后面一个数

//通常与for循环结合使用

1

2

3

⑫遍历vector的三种方法

int main () {

vector<int> a;

for (int i = 0; i < 10; i ++) {

a.push\_back(i);

}

//三种遍历vector的方法

for (int i = 0; i < a.size(); i ++) {

cout << a[i] << ' ';

}

cout << endl;

for (auto i = a.begin(); i != a.end(); i ++) {

cout << \*i << ' ';

}

cout << endl;

//C++11的新语法

for (auto x : a) {

cout << x << ' ';

}

cout << endl;

return 0;

}

⑬结合算法erase() reverse()

#include<algorithm>

a.erase(p)//从a中删除迭代器p指定的元素，p必须指向c中的一个真实元素，不能是最后一个元素end()

a.erase(b,e)//从a中删除迭代器对b和e所表示的范围中的元素，返回e

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

reverse(a.begin(),a.end());//a的值为5，4，3，2，1 倒置

✨2、pair [ x,y ]

可以理解为（x,y）数学中的坐标表示

小技巧：使用typedef定义 typedef pair<int, int> PII

①头文件

#include<utility>

1

②初始化

使用：pair<first数据类型,second的数据类型>元素名；

pair中只有两个元素，first和second。

//俩种方法初始化

pair<string,int> p("hello",1);

p = make\_pair("hello",1);

③first() 和second()

p("hello",1);

p.first; //第一个元素 =hello

p.second; //第二个元素 = 1

④嵌套（套娃）

vector< vector<pair<int, int> > >//与vector结合【再写个vector结合即可】

1

//套娃操作 用pair存储3个数据

pair<int, pair<int, int>> p(1,{2,3});

⑤实战题

可以做下这道题离散化 AcWing 802

⑥当pair 结合 sort()函数使用的时候， pair 默认对first升序，当first相同时对second升序（从小到大）。 也可以通过修改cmp函数达到对second就行排序， 具体可以看这篇博客

✨3、string【字符串】

支持比较操作符>,>=,<,<=,==,!=

①头文件

#include <string>

1

②初始化

string a = "ac";

1

2

③ substr() ！！ 很有用 特别是求子串的时候

#include<iostream>

#include<string>

using namespace std;

int main () {

string a = "ac";

a += "w";//支持比较操作符>,>=,<,<=,==,!=

cout << a << endl; //输出子串a :acw

a += "ing";

cout << a << endl;

//以字符串数组理解

cout << a.substr(0, 3) << endl; //当第一个数是0 则后一位数:输出从头开始的长度为3的子串

cout << a.substr(0, 3) << endl; //当第一个数是1 则输出下标为1 到下标为3的子串

cout << a.substr(0, 9) << endl;//如果超出长度范围 则输出原子串

cout << a.substr(1) << endl; //从下标为1开始输出

cout << a.substr(0) << endl; //原子串

printf("%s\n", a.c\_str());//如果用printf输出

return 0;

}

④ c\_str()

// 返回这个string对应的字符数组的头指针

string s = "Hello World!";

printf("%s", s.c\_str()); //输出 "Hello World!"

1

2

3

⑤push\_back() 和 insert()

// 尾插一个字符

a.push\_back('a');

// insert(pos,char):在制定的位置pos前插入字符char

a.insert(a.begin(),'1'); //输出 1a

//插入字符串

string str2="hello";

string s2="weakhaha";

str2.insert(0,s2,1,3);

//将字符串s2从下标为1的e开始数3个字符，分别是eak，插入原串的下标为0的字符h前

⑥empty()

判断a是否为空，空则返回true，非空则返回false

⑦size() length()

都是 返回字母个数

!!注意他们的类型都是 注：std::string 的成员函数 length() 的返回值类型为 unsigned 类型，因此当 s.length() < t.length() 时，二者相减会得到一个很大的数产生运行时错误，所以相减之前需要先将二者强制类型转换为 int 类型。

做这个题的教训

string s = "cpt";

cout << a.size()<< endl; //输出3

printf("%s", a.length()); //输出 3

⑧clear()

把字符串清空

可以发现string 和vector这些还是有很多共同的函数的

✨4、queue【队列】 和priority\_queue 【优先队列、堆】

队列是一种数据结构 原理：先进先出，元素从一端入队，从另一端出队，就像是排队。

优先队列和队列特性不同：按优先级排序 和 获取

①头文件

#include < queue >//都在这个头文件

1

②初始化

//queue <类型> 变量名

//priority\_queue <类型> 变量名;

queue <int> q; //定义一个名为q队列

priority\_queue <int> q; //默认是大根堆

//定义小根堆

小根堆：priority\_queue <类型,vecotr <类型>,greater <类型>> 变量名

③共同函数

q.size();// 这个队列的长度

q.empty();//用于判断这个队列是否为空，空则返回true，非空则返回false

q.push(); //往队尾插入一个元素

q.pop(); //队列：把队头弹出 优先队列 ：弹出堆顶元素

④区别

队列：

q.front();// 返回队头元素

q.back(); //返回队尾元素

优先队列：

q.top();// 返回堆顶元素

1

⑤清空

注意：队列和堆没有clear函数

所以清空的方法就是重新初始化

q = queue <int> ();

1

✨5、stack 【栈】

①头文件

include<stack>

1

②初始化

//stack<类型> 名字;

stack<int> s;

1

2

③size()

返回这个栈的长度

④push()

向栈顶插入一个元素

⑤top()

返回栈顶元素

⑥pop()

弹出栈顶元素

✨6、deque【双向队列】

好用，几乎其他容器的都有，就是慢一点

①头文件

#include <deque>

1

2

②初始化

deque<int> dq;//定义一个int类型的双向队列

1

③常用函数

dq.size(); //返回这个双端队列的长度

dq.empty(); //返回这个队列是否为空，空则返回true，非空则返回false

dq.clear(); //清空这个双端队列

dq.front(); //返回第一个元素

dq.back(); //返回最后一个元素

dq.push\_back(); //向最后插入一个元素

dq.pop\_back(); //弹出最后一个元素

dq.push\_front(); //向队首插入一个元素

dq.pop\_front();//弹出第一个元素

dq.begin(); //双端队列的第0个数

dq.end(); //双端队列的最后一个的数的后面一个数

✨7、set 【集合】和 multiset

集合与映射也是两个常用的容器,set类似于数学上的集合

①头文件

include<set>

②初始化

set<string> s;//string 集合

1

③区别

set不允许元素重复，如果有重复就会被忽略，但multiset允许.

④常用函数

size();// 返回元素个数

empty(); //返回set是否是空的

clear(); //清空

begin(); //第0个数，支持++或--，返回前驱和后继

end(); //最后一个的数的后面一个数，支持++或--，返回前驱和后继

insert(); //插入一个数

find(); //查找一个数

count(); //返回某一个数的个数

erase(x); //删除所以x 时间复杂度 O(k + logn)

erase(s.begin(),s.end());//删除一个迭代器

⑤核心函数

lower\_bound(x); //返回大于等于x的最小的数的迭代器 核心操作

upper\_bound(x); //返回大于x的最小的数的迭代器 不存在返回end()

1

2

✨8、map 【映射】 /multimap

map就是从键（key）到值（value）的映射。因为重载了[ ]运算符，map像是数组的“高 级版”。例如可以用一个map<string，int>month\_name来表示“月份名字到月份编号”的映射， 然后用month\_name["July"]=7这样的方式来赋值

①头文件

include <map>

1

②初始化

这个初始化有点不同 还是小技巧搭配typedef简化

map<string,int> m = { "A", 10 };

③常用函数

insert(); //插入一个数，插入的数是一个pair

erase();

//（1）输入是pair

//（2）输入一个迭代器，删除这个迭代器

find(); //查找一个数

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

④ 映射 [ ]

时间复杂度 O(logn)

#include <iostream>

#include <string>

#include<map>

using namespace std;

int main()

{

map<string,int>a;

a["abc"] = 1;//把字符串"abc" 映射为1

cout << a["abc"] << endl; //查找abc 程序输出 1

return 0;

}

⑤应用

#include <iostream>

#include <string>

#include<map>

using namespace std;

typedef pair<string,int> PSI;

int main()

{

map<string,int> mp;

mp.insert(make\_pair("heihei",5));

mp.insert(PSI("haha",10));//简化

map<string,int>::iterator it=mp.begin();//迭代器：map<int, char>::iterator it

for(;it!=mp.end();it++)

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

return 0;

}

参考文章

✨9、unordered【哈希表】

这个部分学的不咋样，可以先看看y总的教程,等我会了更新！！

①头文件

unordered\_set，unordered\_map，unordered\_muliset,unordered\_multimap

//头文件就是加上对应名称

1

2

②优势

和上面map 和set类似，增删改查的时间复杂度是O(1)

③缺点

不支持lower\_bound()和upper\_bound()

✨10、bitset 【压位】

它是一种类似数组的结构，它的每一个元素只能是０或１，每个元素仅用１bit空间,用于节省空间，

并且可以直接用01串赋值，可以理解为一个二进制的数组

①头文件

include<bitset>

②初始化

bitset<4> bs;　　//无参构造，长度为４，默认每一位为０

bitset<8> b(12);　　//长度为８，二进制保存，前面用０补充

string s = "100101"; //01串赋值

bitset<10> bs(s);　　//长度为10，前面用０补充

③支持操作

~取反，&与，|与或，^异或

>>，<< 移动

==，!=

[] 取0/1

④常用函数

count(); //返回1的个数

any(); //判断是否至少有一个1

none(); //判断是否全为0

set(); //把所有位置赋值为1

set(k,v); //将第k位变成v

reset(); //把所有位变成0

flip(); //把所有位取反，等价于~

flip(k); //把第k位取反

✨10、Algorithm 【常用算法函数】

没想到吧 ！ 我偷偷更新啦~~ 记录一下 2022.3.28

3.30更新

头文件

#include<algorithm>

以下为常用函数

①、sort();【具有和快排一样的速度】

时间复杂度O (n\*logn)

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

vector<int> b(a,a+5);

sort(a,a+5);//搭配数组 从小到大排序

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

写一个cmp函数 实现从大到小排序

#include <bits/stdc++.h>

using namespace std;

int cmp(int a, int b)

{

return a > b;

//蚂蚁感冒的正负数绝对值 return abs(a) < abs(b);

}

int main () {

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

vector<int> b(a,a+5);

sort(a,a+5);//搭配数组 从小到大排序

sort(b.begin(),b.end());//搭配容器 //从小到大

for (int i = 0; i < 5; i ++) {

cout << a[i] << " ";

}

cout << endl;

for (auto x:b) {

cout << x << " " ;

}

cout << endl;

sort(b.begin(),b.end(),cmp); //从大到小

for (auto x:b) {

cout << x << " ";

}

return 0;

}

②\_\_gcd 最大公约数

最大公约数小题

#include<cstdio>

#include<algorithm>

using namespace std;

int n,m;

int main()

{

scanf("%d %d",&n,&m);

int k=\_\_gcd(n,m);//最大公约数

printf("%d ",k);

printf("%d", n \* m / k); //最小公倍数

return 0;

}

③max min

max(a,b);//返回最大值

min(a,b);//返回最小值

1

2

④swap

swap(a,b);//交换a和b

1

⑤lower\_bound()与upper\_bound() [二分查找]

时间复杂度O(log n)

使用之前一定要先排序

//使用方法

// 练习习题 https://leetcode.cn/problems/zai-pai-xu-shu-zu-zhong-cha-zhao-shu-zi-lcof/?envType=study-plan&id=lcof&plan=lcof&plan\_progress=x5qie1j7

//AC代码

class Solution {

public:

int search(vector<int>& nums, int target) {

int l = lower\_bound(nums.begin(), nums.end(), target) - nums.begin();

int r = upper\_bound(nums.begin(), nums.end(), target) - nums.begin();

return r - l;

}

};

⑥reverse 【倒置】

ector<int> v={1,2,3,4,5};

reverse(v.begin(),v.end());//v的值为5，4，3，2，1 倒置

1

2

⑦find

//在a中的从a.begin()（包括它）到a.end()（不包括它）的元素中查找10，

//若存在返回其在向量中的位置

find(a.begin(),a.end(),10);

⑧、erase【删除】

//从c中删除迭代器p指定的元素，p必须指向c中的一个真实元素，不能等于c.end()

c.erase(p)

//从c中删除迭代器对b和e所表示的范围中的元素，返回e

c.erase(b,e)

✨11、语法小技巧【代码加速器】

①、连续读取

while (cin >> a >> b) { ...}

1

①、加快 cin 和 cout 的速度

ios::sync\_with\_stdio(false);

cin.tie(0);

cout.tie(0);

③、万能头

缺点 耗时

#include <bits/stdc++.h>

④、#define

#define INF 0x3f3f3f3f; //无穷大 10^9

#define x first ;//结合pair

#define y second;

⑤exit(0) [debug]

头文件<cstring> 相当于注释掉

Debug技巧 ： 使用exit(0) 中断程序，如果没出现问题，则继续往下exit(0),直到发现问题，则Debug在附近

通常用于调试 无输出的

⑥无穷大

0x3f3f3f3f的十进制为1061109567，和INT\_MAX一个数量级，即109数量级，而一般场合下的数据都是小于109的。

0x3f3f3f3f \* 2 = 2122219134，无穷大相加依然不会溢出。

可以使用memset(array, 0x3f, sizeof(array))来为数组设初值为0x3f3f3f3f，因为这个数的每个字节都是0x3f。

⑦ next\_permutation （全排列）

作用：

将当前排列更改为全排列中的下一个排列。

如果当前排列已经是全排列中的最后一个排列（元素完全从大到小排列），函数返回 false 并将排列更改为全排列中的第一个排列（元素完全从小到大排列）；否则，函数返回 true。

// 1 结合 数组

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

do{

for(int i = 0; i < 5; i ++) cout << a[i] << " ";

cout << endl;

}while(next\_permutation(a, a + 5));

// 2结合 vector

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

do{

for(int i = 0; i < a.size(); i ++) cout << a[i] << " ";

cout << endl;

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

————————————————

版权声明：本文为CSDN博主「CPT1024」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_49486457/article/details/123439229