STL,基础数学

算法	类型	Structure	function	备注
不变序列算法O(n)	顺序容器/关联容器均适用		iterator min_element(iterator first, iterator last[,Pred op])	
			iterator max_element(iterator first, iterator last[,Pred op])	
			int count(iterator first, iterator last,const T& val)	
			iterator find(iterator first, iterator last,const T& val)	
插入算法第一个被插入元素的迭代器first	Single element	vector	iterator insert (const_iterator position, const value_type& val);	
		set	pair<iterator,bool> insert (const value_type& val);	pair::second inserted(1)or existed(0)
		String	string& insert (size_t pos, const char* s);	
	Fill	vector	iterator insert (const_iterator position, size_type n, const value_type& val);	
		String	string& insert (size_t pos, size_t n, char c);	
	Range	vector,String	iterator insert (const_iterator position, InputIterator first, InputIterator last);	
	Buffer	String	string& insert(size_t pos, const string& str, size_t subpos, size_t sublen);	个数
删除算法返回被删除元素的下一个迭代器last	Single element	vector,set	iterator erase (const_iterator position);	
	Unfill	set	size_type erase(const value_type& val);	个数
	Range	vector,set	iterator erase(const_iterator first, const_iterator last);	

1.输入方式大总结

> getline(cin,str):
> std::basic_istream& getline(std::basic_istream&__is, std::__cxx11::basic_string& __str);

```
//解析一个字符的三种方式
scanf("%c",&ch);
ch=cin.get();
ch=getchar();
//解析一个字符串的方式
char m[20];
gets(m);
cin.getline(m,5);//多了一个参数，可以加结束符
cin.get(m,20);//多了一个参数，可以加结束符
```

> %2d就是将数字按照宽度为2 采用右对齐方式输出，若数据位数不到2位，则左边补空格

> %02d和%2d差不多 只不过左边补0

> toupper,tolower的使用

2."对应"的处理

法1:建表

```
char c[14] = {"0123456789ABC"};
printf("#");
for(int i = 0; i < 3; i++) {
    int num;
    scanf("%d", &num);
    printf("%c%c", c[num/13], c[num%13]);
}
return 0;
```

法2:直接函数输出

3.cmp的写法

```
int cmp(node& a, node& b) {//Tips:引用传递和地址传递效率高于直接传递
        if ((a.de1 + a.cai) != (b.de + b.cai))
        return (a.de + a.cai) > (b.de + b.cai);//技巧:以求和代替求平均值
        else if (a.de != b.de) return a.de > b.de;
        else return strcmp(a.name, b.name) < 0;//char name[9]
        //还可以string a.name<b.name;
}
```

> 并列排名问题

```
struct node {
        int id, best;
        int score[4], rank[4];
```

```
};//一般一个结构体
for(i=0;i<sch.size();++i){
 for(j=0;j<k;++j){
        bool y;
        int choice=sch[i].sc[j];
        int x=ans[choice].size();
        if(x>0){
            int last=x-1;
            y=sch[i].grade[0]==ans[choice]
[last].grade[0]&&sch[i].grade[1]==ans[choice][last].grade[1];
            x=y?ans[choice][last].rank:x;
        }
        if(x<full[choice]||y){
            ans[sch[i].sc[j]].push_back(sch[i];
            break;
        }
}
for(flag = 0; flag <= 2; flag++) {
        sort(ans.begin(), ans.end(), cmp1);
        for(int i = 0; i < n; i++) {
            ans[i].rank[flag] = i;
            if(i>0&&ans[i].grade[flag] == ans[i-1].grade[flag])//注意多一个判断还有
反向
            ans[i].ans[flag] = ans[i-1].rank[flag];
        }
}
```

4.输出格式

> 普通输出问题

```
for (int i = 0; i < 4; i++) {
        sort(v[i].begin(), v[i].end(), cmp);
        for (int j = 0; j < v[i].size(); j++){//小技巧:利用顺序条件，如求一串码中只出
现一次的元素
        if(j!=0)printf(" ");
                printf("%d", v[i][j].num);
    }
    printf("\n");
}
```

> 条件补零问题Eg:保留n位小数(temp.size可大可小)

```
while(temp.size()<N)//有效数字位数小于N
temp+="0";//在字符串末尾加足够的0保证有N位有效数字
```

5.排序题

> 多维排序-->找参考系

1比较时间实际可以t[j]-t[i](从0:0:0开始)

2求连续字符串和满足一定条件-->sum[j]-sum[i]

[易错点]多余链表、记录的问题

PAT1016 Phone Bills

> 夹逼题:[a,b]!c

```
for(j=temp;st[j].time<=cal(c)&&j<st.size();++j)
    if(st[j].status==0)sum++;
    else sum--;
```

> 堆

```
//[f,l),[,Compare comp]
void pop_heap(iterator first, iterator last);//向下调整,原堆顶->last-1,O(log(n))
void push_heap(iterator first, iterator last);//向上调整O(log(n))
void make_heap(iterator first, iterator last);//建堆O(n)
void sort_heap(iterator first, iterator last);//建堆后排序O(nlog(n))
```

6.运行超时:

- 循环内套了一个大函数:如循环内sort(>1000),或是一个大数组(10^5)的cin-->scanf
- 外循环与内循环条件对换，可以减少重复