

入队杂谈 & 基础复习

庄博伟

ZUCC ACM Group

January 17th 2022



目录

- 1 集训安排
- 2 Codeforces 快速入门
- 3 XCPC 杂谈
- 4 天梯赛
- 5 新生赛复盘 & 复习



集训时间

大致安排如下，详细内容见[腾讯文档](#)

日期	安排
1/17	入队杂谈 & 基础复习
1/18	dp 入门
1/20	树基础
1/21	图论入门 & 最短路
1/23	常见套路杂讲
1/25	最小生成树
1/27	简单数论
1/28	字符串
2/10	测试 1
2/12	测试 2
2/14	测试 3
2/16	测试 4



集训要求

- 讲课的安排时间为当天 10:00 - 18:00, 12:00 - 14:00 午休, 形式由主讲人自行安排
- 测试的安排时间为当天 13:00 - 17:00, 测试要求全程录屏, 录屏软件和录屏要求会在测试前一天发放
- 有事无法参加当天集训需要提前向章晨榆请假, 讲课录播会传到 B 站, 可以看录播



集训 QQ 群

集训相关通知都会通过该群发送



集训目标

本次集训的基础目标是所有人都能**熟练掌握机房试训所教授的全部内容**，并在寒假的新内容上有不同程度的收获

正式入队要求

四次测试综排在前 30 (含 ZUCC 和 HZNU 所有新生)，落选的话在今年暑假开始前 Codeforces 能达到 1500 分也可以正式入队



目录

- 1 集训安排
- 2 Codeforces 快速入门
- 3 XCPC 杂谈
- 4 天梯赛
- 5 新生赛复盘 & 复习



CF 介绍

Codeforces 是 XCPC 选手最常用的训练平台之一，有着大量的优质题目和优质比赛

比赛表现会影响选手的 rating，通过寒假集训入队的大一选手要求在暑假开始前**达到过**1400 分（最高分 ≥ 1400 ）

Candidate Master
Kyooma ★
Hangzhou, China
From Zhejiang University City College

Contest rating: **1918** (max. candidate master, 1918)

Contribution: 0

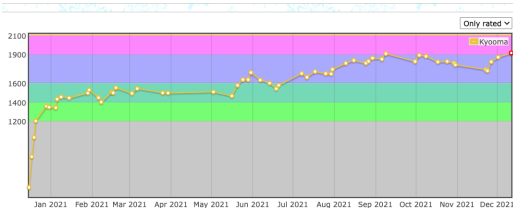
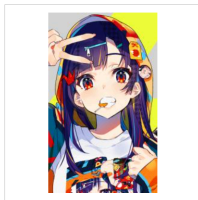
Friend of: 55 users

Last visit: 31 hour(s) ago

Registered: 14 months ago

[Comments](#)

[Talks](#) | [Send message](#)






CF 比赛介绍

- CF 主要有 2 种赛制：CF 赛制和 ICPC 赛制，4 种比赛：div1, div2, div3 和 Edu Round
- 大部分比赛时间为晚上 10:35 开始，时长一般 2 小时出个头，题目风格偏思维



CF 赛制

Standings									
#	Who	=	*	A 500	B 1000	C 1500	D 2250	E 3000	F 3250
1	 zero4338	8048		492 00:04	952 00:12	1314 00:31	1737 00:57	1584 01:58	1969 01:27
2	C2020jzm	7221		496 00:02	914 00:09	1422 00:13	1989 00:29	2400 00:50	
3	SYDevil	7037		494 00:03	972 00:07	1392 00:18	1903 00:33	2276 00:52	
4	 PPL_	6895	+1	480 00:10	894 00:14	1368 00:22	1845 00:45	2208 01:06	
5	 just_ice	6865		498 00:01	972 00:07	1440 00:10	2043 00:23	1912 01:14	
6	dongruixuan	6734		492 00:04	976 00:06	1348 00:17	1844 00:34	2074 01:13	

- 不同难度的题目分值不同，并且解的越快错误提交越少，分值越高
- 每道题的测试数据只会给出部分，在比赛结束后，才会用完整的测试数据重跑所有提交，只有通过了完整数据才能在结算比赛时获得本题分数，否则本题算不通过，也就是 FST (Failed System Test)
- 选手在赛中可以互相 HACK 提交，被 HACK 的一方可以重新提交，有补救的机会



- 大部分 CF 比赛中 FST 和 HACK 是很少出现的，因为题目的部分数据基本足够强
- 不要瞎 HACK 别人，HACK 失败是会扣分的，新手上路只管写自己的题就行了，被 HACK 了要及时补救
- CF 几乎所有比赛的题目都是按照难度排序的，按顺序做题是最稳健的拿分方式
- 自己遇到 FST 或许 HACK 的话好好反省，是思路问题还是算法理解问题还是代码力问题








ICPC 赛制

ICPC 赛制就是新生赛时候的赛制，数据完整，题目分值相同均为 1

不过 CF 的 ICPC 赛制和常规的 ICPC 赛制有小部分区别

- 每发错误提交罚时只有 5 分钟
- 虽然数据完整，但比赛结束后有 12 小时的 opening HACK 时间，由选手来互相 HACK，因此还是可能出现 FST

Standings										
#	Who	=	Penalty	*	A	B	C	D	E	F
1	star0-0	6	202		+ 00:02	+ 00:04	+ 00:05	+3 00:44	+ 00:12	+2 01:25
2	 Heltion	6	203		+ 00:04	+ 00:06	+ 00:14	+ 00:57	+ 00:36	+ 01:26
3	 Geothermal	6	216		+ 00:01	+ 00:02	+ 00:06	+1 01:40	+ 00:22	+1 01:05
4	 neal	6	219		+ 00:02	+ 00:04	+ 00:07	+1 00:54	+ 00:50	+ 01:32
5	 imeimi	6	221		+ 00:02	+ 00:07	+ 00:13	+1 00:42	+ 00:59	+ 01:28
6	 m_99	6	229		+ 00:02	+ 00:05	+ 00:11	+1 00:57	+ 00:34	+ 01:50



CF 比赛介绍

CF 主要有

- div1: CF 赛制, 难度高, 仅允许 rating1900 及以上参加
- div2: CF 赛制, 难度适中, 所有人可参加但仅 2100 以下计分
- div3: ICPC 赛制, 难度低, 所有人可参加但仅 1600 以下计分
- Edu: ICPC 赛制, 难度类似 div2, 所有人可参加但仅 2100 以下计分

关于难度

除了这些, 还有一些特殊场次, 但除了 div1, 几乎所有场次都有 2 道及以上比较简单的题 (AB), 新手直接打 div2 或者 Edu 是完全没有问题的



注意事项

- 不要开小号，不要怕掉分，2 小时的比赛掉点分都扛不住，区域赛 5 小时不得直接噶了？
- 不要开黑，不要在比赛结束前交流题意或做法，一经发现直接开除
- 不要开网页翻译，有个学长经常开翻译，后遗症是正赛天天读错题
- 如果英语不好，可以翻译器仅翻译单个词汇，当词典用。而且竞赛常见词汇也就那几个，因此才说英语水平不影响打竞赛



如何参加 Codeforces 比赛



寒假讲课配套的练习也在 CF 上，需要加入 Group，发现自己 CF 没有加入这个 Group 的 QQ 私聊我

ZAWEI SETTINGS LISTS BLOG TEAMS SUBMISSIONS FAVOURITES **GROUPS** CONTESTS PROBLEMSSETTING

 [All groups](#)  [Create group](#)

Zawei's groups

Group name	Role	Invitation	Member since	Invited on	
ZUCC & HZNU Winter Camp 2022	Manager	Accepted	Jan/16/2022 15:20 ^{UTC+8}	Jan/09/2022 11:12 ^{UTC+8}	



目录

- 1 集训安排
- 2 Codeforces 快速入门
- 3 XCPC 杂谈
- 4 天梯赛
- 5 新生赛复盘 & 复习



大致赛程

- ICPC、CCPC 基本都在下半年，有多场区域赛，ACM 牌子基本是指这两个比赛的牌子，难度高，认可度高
- 往前推是暑假，暑假留校集训，强度高难度大，区域赛名额根据暑期集训分配
- 再往前推，每年的上半年是比较空闲的时候，比赛少，比较重要的有省赛和天梯赛，前者类似 XCPC 比赛，但是更好获奖一点，后者一会儿单独讲



将来的目标

省赛的奖并不是很顶用，因此这里只谈 XCPC 赛事

- 回报上讲，银牌及以上是比较顶用的，铜牌很多场合差点意思，给自己定目标的话起码定到银牌及以上
- 难度上讲，3 个 CF1900 可以保银，1 个 1900+2 个 1600 可以争银
- 金牌基本需要 3 个 CF2100 或者 1 个 CF2400



今年的目标

- 省赛只有 1~2 支新生队伍有参赛名额
- 浙江省赛的难度是很有参考价值的，在暑假开始前练习省赛，如果能拿到铜中游及以上，那么今年区域赛直接拿奖是很有希望的，可以先以此为小目标来练习



补题很重要，补好一场比赛比新开一场比赛更重要

- 补题并非越多越难越好，写超出自己能力范围太多的题目不会有太好的效果
- 补题应当和自己目前的目标同步，比如当前目标是拿个银牌，那就应该补到银牌区为止
- CF 的补题同理：蓝名需要在 div2 稳定出 3 题及以上，如果以蓝名为目标，那补题应该至少补到 3 题



目录

- 1 集训安排
- 2 Codeforces 快速入门
- 3 XCPC 杂谈
- 4 天梯赛**
- 5 新生赛复盘 & 复习



天梯赛介绍

- 表面团体赛，实则个人赛，10 人一队，队伍得分为队员的总分，题目难度从低到高分为 L1, L2, L3
- 题目有部分分，正赛时可以骗分，比赛平台在 PTA



寒假作业

PTA 上有历年天梯赛的[题库](#)，正式入队的新生需要达到以下要求

- L1 刷到 60 题及以上
- L2 刷到 30 题及以上
- 拿到满分的题目才算有效题数



PTA 上有历年天梯赛的[题库](#)，正式入队的新生需要达到以下要求

- L1 刷到 60 题及以上
- L2 刷到 30 题及以上
- 拿到满分的题目才算有效题数

知识点

L1 与 L2 的题目除少数几题外，不会涉及高深算法，基本偏重于基础数据结构和 STL 的应用，对于提升基础和码力很有用

对 STL 掌握还不是很好的我推荐看看[BIT 的冬训 STL 专题](#)



目录

- 1 集训安排
- 2 Codeforces 快速入门
- 3 XCPC 杂谈
- 4 天梯赛
- 5 新生赛复盘 & 复习



看了下大伙的新生赛代码



[新生赛补题链接](#)，题面也在里面，PPT 里面就不插题面了



Problem B. Boboge and Tall Building

- 这个题做法没啥好讲的，主要是一个精度问题
- 题目的输出格式中有对输出的以下要求

Output

For each test case, output the height of the floor Boboge lives in. Your answer will be accepted if absolute or relative error does not exceed 10^{-6} . Formally, let your answer be a , and the jury's answer be b . Your answer is considered correct if $\frac{|a-b|}{\max(1, |b|)} \leq 10^{-6}$.



Problem B. Boboge and Tall Building

- 这个题做法没啥好讲的，主要是一个精度问题
- 题目的输出格式中有对输出的以下要求

Output

For each test case, output the height of the floor Boboge lives in. Your answer will be accepted if absolute or relative error does not exceed 10^{-6} . Formally, let your answer be a , and the jury's answer be b . Your answer is considered correct if $\frac{|a-b|}{\max(1, |b|)} \leq 10^{-6}$.

翻译成人话就是绝对误差和相对误差至少要有有一个满足 $\leq 10^{-6}$

- 绝对误差: $|b - a|$
- 相对误差: $\frac{|b-a|}{|b|}$
- 上面的式子等价于 $\min(|b - a|, \frac{|b-a|}{|b|}) \leq 10^{-6}$



Problem B. Boboge and Tall Building

- 这个题做法没啥好讲的，主要是一个精度问题
- 题目的输出格式中有对输出的以下要求

Output

For each test case, output the height of the floor Boboge lives in. Your answer will be accepted if absolute or relative error does not exceed 10^{-6} . Formally, let your answer be a , and the jury's answer be b . Your answer is considered correct if $\frac{|a-b|}{\max(1, |b|)} \leq 10^{-6}$.

翻译成人话就是绝对误差和相对误差至少要有有一个满足 $\leq 10^{-6}$

- 绝对误差: $|b - a|$
- 相对误差: $\frac{|b-a|}{|b|}$
- 上面的式子等价于 $\min(|b - a|, \frac{|b-a|}{|b|}) \leq 10^{-6}$

在这题中，只要保证小数点后至少输出 6 位，就可以一直满足绝对误差

为什么会有相对误差？

- 相对误差在这题中用不上，这题这么写是因为，竞赛中涉及浮点的题目一般都使用 $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-x}$ 的形式来要求精度
- 那么相对误差有什么用呢？

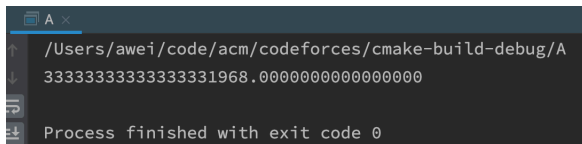


为什么会有相对误差？

当绝对误差无法保证满足的时候，就需要相对误差了。

- double 虽然可以存的权值范围很大，但它只能存储 16 位有效数字
- 最简单的例子

```
1 double x = 1e20; // x = 1020
2 printf("%.16f\n", x / 3);
```



```
A x
/Users/awei/code/acm/codeforces/cmake-build-debug/A
33333333333333331968.0000000000000000
Process finished with exit code 0
```

这种事绝对很奇怪啊.jpg



为什么会有相对误差？

- 需要的有效数字位数太多的时候，绝对误差是无法保证的
- 相对误差意味着你只需要保证前几位有效数字相同就行了
- 以本题 10^{-6} 的相对误差为例，我们只要保留前 7 位数字（四舍五入）就可以了



为什么会有相对误差？

- 需要的有效数字位数太多的时候，绝对误差是无法保证的
- 相对误差意味着你只需要保证前几位有效数字相同就行了
- 以本题 10^{-6} 的相对误差为例，我们只要保留前 7 位数字（四舍五入）就可以了

为了保证绝对误差，我们输出了小数点前至少 1 位 + 小数点后 6 位，自动满足相对误差至少 7 位的要求，所以**实际写题时可以只考虑绝对误差来写**



小结

- 当要求精度 $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-x}$ 时, 至少输出 x 位小数



● 来看份 B 题的 WA 代码

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int t;
6     cin>>t;
7     while(t--){
8         int m,n,k;
9         cin>>n>>m>>k;
10        cout<<(k*1.0/m)*(n-1)<<'\\n';
11    }
12 }
```



- 来看份 B 题的 WA 代码

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int t;
6     cin>>t;
7     while(t--){
8         int m,n,k;
9         cin>>n>>m>>k;
10        cout<<(k*1.0/m)*(n-1)<<'\n';
11    }
12 }
```

- cout 默认的输出浮点数方式是输出前 6 位有效数字，如 24.123456，会输出 24.1235，因此既不满足绝对误差也不满足相对误差，寄了

仅作为我个人建议：少用 cin 和 cout，多用 scanf 和 printf



战犯代码插播

```
1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 double n,m,k;
6 double ans;
7
8 int solve(){
9     cin>>n>>m>>k;
10    printf("%lf\n",k/m*(n-1));
11 }
12
13 int main(){
14     int t;
15     cin>>t;
16     while(t--){
17         solve();
18     }
19 }
```



战犯代码插播

```
1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 double n,m,k;
6 double ans;
7
8 int solve(){
9     cin>>n>>m>>k;
10    printf("%lf\n",k/m*(n-1));
11 }
12
13 int main(){
14     int t;
15     cin>>t;
16     while(t--){
17         solve();
18     }
19 }
```

- `int solve()` 没有返回值，有返回值的函数不返回内容是未定义行为
- `main()` 中不加 `return 0` 也不是好习惯，但大部分情况不影响评测，某些比赛或者平台似乎会影响（如蓝桥杯），最好还是加上

未定义行为

未定义行为指的是逻辑上不合法的操作，不同版本的编译器对未定义行为有不同的编译结果，也就是可能本地过样例了，OJ 上样例都没过，或者换个 OJ 就过了

- 有返回值的函数都加上返回值



Problem D. Diseased String

这题做法也没啥好讲的，来看一份 RE 的代码

```
9  int cnt=0;
10 int i=0;
11 while(i<str.size()-2&&str.find("y",i)!=-1)
12 {
13     if(str.at(str.find("y",i)+1)=='b'&&str.at(str.find("y",i)+2)=='b')
14     {
15         int x=3;
16         cnt++;
17         while(str.find("y",i)+x<str.size()&&str.at(str.find("y",i)+x)=='b')
18         {
19             cnt++;
20             x++;
21         }
22     }
23     i=str.find("y",i)+1;
24 }
25 cout<<cnt<<endl;
```



Problem D. Diseased String

这题做法也没啥好讲的，来看一份 RE 的代码

```
9  int cnt=0;
10 int i=0;
11 while(i<str.size()-2&&str.find("y",i)!=-1)
12 {
13     if(str.at(str.find("y",i)+1)=='b'&&str.at(str.find("y",i)+2)=='b')
14     {
15         int x=3;
16         cnt++;
17         while(str.find("y",i)+x<str.size()&&str.at(str.find("y",i)+x)=='b')
18         {
19             cnt++;
20             x++;
21         }
22     }
23     i=str.find("y",i)+1;
24 }
25 cout<<cnt<<endl;
```

- 这份代码 RE 在第一个 if 上，`str.find("y", i)` 是寻找从 i 开始的一个 `y` 的位置，但是如果 `y` 处在字符串最后面，会导致数组越界
- 这份代码还有个小问题：`str.find()` 重复调用太多次了，可以开一个本地变量存储 `str.find("y", i)` 的值，虽然不影响整体时间复杂度，但是常数会小一些



Problem G. Generate 7 Colors

- 这题做法还是值得讲一讲，[题解](#)



Problem G. Generate 7 Colors

- 这题做法还是值得讲一讲, [题解](#)
- 另外挑了一份战犯代码, 评测结果是 TLE

```
5  const int N=1e7;
6  int a[N];
7  void solve(){
8      for(int i=0;i<7;i++)cin>>a[i];
9      for(int i=0;i<6;i++){
10         if(a[i+1]>a[i]){
11             cout<<-1<<'\n';
12             return;
13         }
14     }
15     int ans=0;
16     for(int i=6;i>=0;i++){
17         if(a[i+1]!=a[i])ans+=a[i]-a[i+1];
18     }
19     cout<<ans<<'\n';
20 }
```



Problem G. Generate 7 Colors

- 这题做法还是值得讲一讲，[题解](#)
- 另外挑了一份战犯代码，评测结果是 TLE

```
5  const int N=1e7;
6  int a[N];
7  void solve(){
8      for(int i=0;i<7;i++)cin>>a[i];
9      for(int i=0;i<6;i++){
10         if(a[i+1]>a[i]){
11             cout<<-1<<"\n";
12             return;
13         }
14     }
15     int ans=0;
16     for(int i=6;i>=0;i++){
17         if(a[i+1]!=a[i])ans+=a[i]-a[i+1];
18     }
19     cout<<ans<<"\n";
20 }
```

- TLE 的原因在于下面的 $i--$ 写成了 $i++$
- 还有个不好的细节，虽然不影响答案，就是最后一个 for 应当从 $\text{int } i = 5$ 开始



Problem H. Hile and Subsequences' MEX

- 这题做法也值得讲一讲, [题解](#)



Problem H. Hile and Subsequences' MEX

- 这题做法也值得讲一讲, [题解](#)
- 战犯环节, 评测结果是 TLE

```
7 void solve(){
8     ans+=n;
9     long long qs=1;
10    for(int i=1;i<=n-1;i++){
11        ans=(ans+qs*(n-i))%998244353;
12        qs=qs*2%998244353;
13    }
14    cout <<ans<<'\n';
15 }
```



Problem H. Hile and Subsequences' MEX

- 这题做法也值得讲一讲, [题解](#)
- 战犯环节, 评测结果是 TLE

```
7 void solve(){
8     ans+=n;
9     long long qs=1;
10    for(int i=1;i<=n-1;i++){
11        ans=(ans+qs*(n-i))%998244353;
12        qs=qs*2%998244353;
13    }
14    cout <<ans<<'\n';
15 }
```

- 单组时间复杂度为 $O(N)$, 本题 $N1e9$, 甚至还有 $T1e5$ 组, 最坏复杂度 $O(NT)$, $1e14$

时间复杂度

评测机每秒能跑的指令数大约在 $2e8$, 实际算法一定存在常数, 以 1 秒的题为例, 程序整体算法时间复杂度不应该超过 $1e8$, 最好不要超过 $5e7$



战犯代码插播

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 long long fast_pow(int a,int n){
4     long long ans=1;
5     for (int i=1;i<=n;i++){
6         ans=ans*a%998244353;
7     }
8     return ans;
9 }
10
11 long long solve(int k){
12     long long sum=((1+k)*k)/2;
13     for (int i=1;i<=k-2;i++){
14         long long count=fast_pow(2,k-(i+1))-1;
15         long long ans=i*count%998244353;
16         sum+=ans;
17     }
18     return sum%998244353;
19 }
20 int main(){
21     int n;
22     cin >> n;
23     while (n--){
24         int k;
25         cin >> k;
26         if (k==1000000000){
27             cout << "851104390" << '\n';
28         }
29         else{
30             cout << solve(k) << '\n';
31         }
32     }
33 }
34 }
```



战犯代码插播

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 long long fast_pow(int a,int n){
4     long long ans=1;
5     for (int i=1;i<=n;i++){
6         ans=ans*a%998244353;
7     }
8     return ans;
9 }
10
11 long long solve(int k){
12     long long sum=((1+k)*k)/2;
13     for (int i=1;i<=k-2;i++){
14         long long count=fast_pow(2,k-(i+1))-1;
15         long long ans=i*count%998244353;
16         sum+=ans;
17     }
18     return sum%998244353;
19 }
20 int main(){
21     int n;
22     cin >> n;
23     while (n--){
24         int k;
25         cin >> k;
26         if (k==1000000000){
27             cout << "851104390" << '\n';
28         }
29         else{
30             cout << solve(k) << '\n';
31         }
32     }
33 }
34 }
```

- 首先 main() 没有 return 0
- 然后这个 fast_pow(), fast 了但没完全 fast
- 还有这个 if k=1000000000, 这招在有部分分的比赛（天梯赛）可以用



Very Useful Algorithm — 打表

- 关于 H 题还有些别的想讲的



Very Useful Algorithm — 打表

- 关于 H 题还有些别的想讲的
- 首先, 如果只能推出 $f_n = n + \sum_{i=0}^{n-1} i \times 2^{n-1-i}$, 但不知道如何化简, 完全可以先用程序跑出 f_n 的前几项, 对着找规律, 这个就叫打表



Very Useful Algorithm — 打表

- 关于 H 题还有些别的想讲的
- 首先, 如果只能推出 $f_n = n + \sum_{i=0}^{n-1} i \times 2^{n-1-i}$, 但不知道如何化简, 完全可以先用程序跑出 f_n 的前几项, 对着找规律, 这个就叫打表
- 哪怕 $f_n = n + \sum_{i=0}^{n-1} i \times 2^{n-1-i}$ 都没有推出来, 你也可以 $O(2^n)$ 枚举每个数字选或者不选, 来暴力打表



Very Useful Algorithm — 打表

- 关于 H 题还有些别的想讲的
- 首先, 如果只能推出 $f_n = n + \sum_{i=0}^{n-1} i \times 2^{n-1-i}$, 但不知道如何化简, 完全可以先用程序跑出 f_n 的前几项, 对着找规律, 这个就叫打表
- 哪怕 $f_n = n + \sum_{i=0}^{n-1} i \times 2^{n-1-i}$ 都没有推出来, 你也可以 $O(2^n)$ 枚举每个数字选或者不选, 来暴力打表

Try a Try, AC is OK

当你发现题目的答案是 $[1, 3, 7, 15, 31, 63, \dots]$ 的时候, 心中一定会有个大胆的猜想吧



- 做法没啥好讲的，借这题讲一下字符串的相关操作



● 做法没啥好讲的，借这题讲一下字符串的相关操作

```
3 char s[11][100]={{'N','e','w','b','i','e'},{'P','u','p','i','l'},{'S','p','e','c','i','a','l','i','s','t'},
4 {'E','x','p','e','r','t'},{'C','a','n','d','i','d','a','t','e',' ','m','a','s','t','e','r'},
5 {'M','a','s','t','e','r'},{'I','n','t','e','r','n','a','t','i','o','n','a','l',' ','m','a','s','t','e','r'},
6 {'G','r','a','n','d','m','a','s','t','e','r'},{'I','n','t','e','r','n','a','t','i','o','n','a','l',' ','g','r','a','n'},
7 {'L','e','g','e','n','d','a','r','y',' ','g','r','a','n','d','m','a','s','t','e','r'}};
```



字符串赋值

整段字符串的赋值方式

- `char[]` 在声明的时候可以整段赋值，其余时候不可以
- `string` 任何时候都可以整段赋值

```
1 char s[11][20] = {  
2     "123",  
3     "456",  
4     "789"  
5 };  
6 s[0] = "123"; // 非法操作  
7 string ss[10] = {  
8     "123",  
9     "234"  
10 };  
11 ss[0] = "789";
```



字符串赋值

- string 和 vector 一样，内存是动态的，因此以下用法会导致数组越界

```
1 string s;  
2 s[0] = 'A';
```



字符串赋值

- string 和 vector 一样，内存是动态的，因此以下用法会导致数组越界

```
1 string s;  
2 s[0] = 'A';
```

- 正确用法是使用 += 或者 resize，类似 vector 的 push_back 和 resize

```
1 string s;  
2 s += 'A';  
3 s.resize(10);  
4 s[2] = 'C';  
5 s[1] = 'B';
```



字符串输入

两个常用的读入方式：按词读入，按行读入



字符串输入

两个常用的读入方式：按词读入，按行读入

- scanf("%s", s); 会从接下来第一个有效字符开始读入，直到碰到无效字符为止，无效字符包含空格，\n，\t 等等，一般用于按词读入

```
1 char s[10];  
2 scanf("%s", s);
```



字符串输入

两个常用的读入方式：按词读入，按行读入

- `scanf("%s", s);` 会从接下来第一个有效字符开始读入，直到碰到无效字符为止，无效字符包含空格，`\n`，`\t` 等等，一般用于按词读入

```
1 char s[10];  
2 scanf("%s", s);
```

- `scanf("%[\\n]", s);` 会一直读入直到碰到 `\n` 为止，一般用于按行读入，注意这个读入方式不会读入 `\n` 本身，需要用 `getchar()` 处理 `\n`，否则读入会一直卡住

```
1 int T;  
2 scanf("%d", &T);  
3 while (T--) {  
4     char s[20];  
5     getchar();  
6     scanf("%[\\n]", s);  
7 }
```

Example. 读入 T 行字符串



char[] 与 string 转化

前面讲了一下少用 cin 和 cout, 那么不用 cin 和 cout 怎么输入输出 string 呢



char[] 与 string 转化

前面讲了一下少用 cin 和 cout, 那么不用 cin 和 cout 怎么输入输出 string 呢

- 输入: 输入到 char[], 然后直接等号赋值给 string 就行了

```
1 char tmp[10];  
2 scanf("%s", tmp);  
3 string s = tmp;
```



char[] 与 string 转化

前面讲了一下少用 cin 和 cout, 那么不用 cin 和 cout 怎么输入输出 string 呢

- 输入: 输入到 char[], 然后直接等号赋值给 string 就行了

```
1 char tmp[10];  
2 scanf("%s", tmp);  
3 string s = tmp;
```

- 输出: string 有个成员函数叫 c_str(), 返回类型为 const char*

```
1 string s = "123";  
2 printf("%s\n", s.c_str());
```



char[] 与 string 转化

前面讲了一下少用 cin 和 cout, 那么不用 cin 和 cout 怎么输入输出 string 呢

- 输入: 输入到 char[], 然后直接等号赋值给 string 就行了

```
1 char tmp[10];  
2 scanf("%s", tmp);  
3 string s = tmp;
```

- 输出: string 有个成员函数叫 c_str(), 返回类型为 const char*

```
1 string s = "123";  
2 printf("%s\n", s.c_str());
```

- string 转 char[]: char[] 是不能直接赋值的, 需要用 strcpy 这个函数

```
1 string s = "123";  
2 char t[10];  
3 strcpy(t, s.c_str());
```



再讲个比较常用的 STL 吧, sort 函数



再讲个比较常用的 STL 吧, sort 函数

- `int[]` 排序, $a[0, 1, 2, \dots, n - 1]$ 从小到大排序

```
1 sort(a, a + n);
```



再讲个比较常用的 STL 吧, sort 函数

- `int[]` 排序, $a[0, 1, 2, \dots, n - 1]$ 从小到大排序

```
1 sort(a, a + n);
```

- `vector` 排序, $vec[0, 1, 2, \dots, n - 1]$ 从小到大排序

```
1 sort(vec.begin(), vec.begin()+n);
```



结构体排序

```
1 struct Student {  
2     char name[20];  
3     double avg;  
4     int score;  
5     int id;  
6     bool operator < (const Student &tmp) const {  
7         if (score == tmp.score) return id < tmp.id;  
8         return score < tmp.score;  
9     }  
10 } stu[107];  
11  
12 sort(stu, stu + 100);  
13 if (stu[0] < stu[1]) {  
14     printf("0<1");  
15 }
```

重载结构体的 `<` 符号后就可以直接 `sort` 了，也可以直接进行 `stu[0] < stu[1]` 之类的逻辑运算



结构体排序

```
1 struct Student {  
2     char name[20];  
3     double avg;  
4     int score;  
5     int id;  
6     bool operator < (const Student &tmp) const {  
7         if (score == tmp.score) return id < tmp.id;  
8         return score < tmp.score;  
9     }  
10 } stu[107];  
11  
12 sort(stu, stu + 100);  
13 if (stu[0] < stu[1]) {  
14     printf("0<1");  
15 }
```

重载结构体的 `<` 符号后就可以直接 `sort` 了，也可以直接进行 `stu[0] < stu[1]` 之类的逻辑运算

重载 `<` 符号的过程不用完全理解也没关系，会举一反三就行了，比如要改成第一关键字按 `avg` 升序，第二关键字按 `score` 升序，第三关键字按 `id` 升序，要怎么改。写法大不了先背一下。

- 学 STL 的最好方式还是多用，碰到不会的再去看资料查一查，推荐资料[BIT 的冬训 STL 专题](#)，[黑马程序员 C++ 教程](#)（P40 开始，前面的不用看）
- 第一天不布置配套练习，可以摸一下 CF 和天梯赛题目集
- 几个常用容器（vector, string, stack, queue, map, set, priority_queue）的常用操作玩一玩足够应付天梯赛大部分题了



END

END

