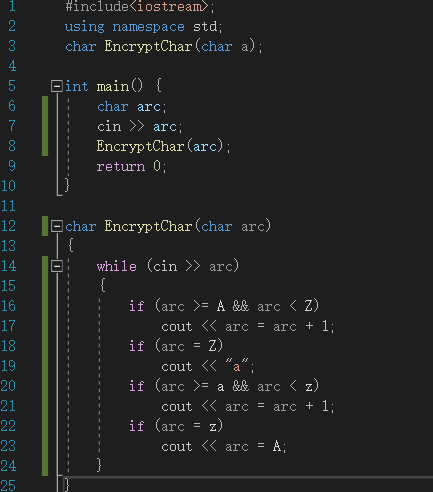
11月23日夜记梦

1. 没打using namespace std；(补图)

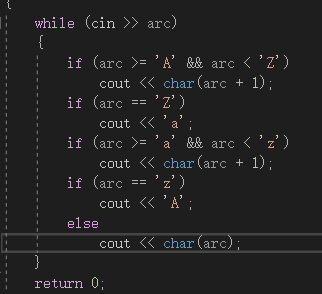
导致cin cout endl闪红；

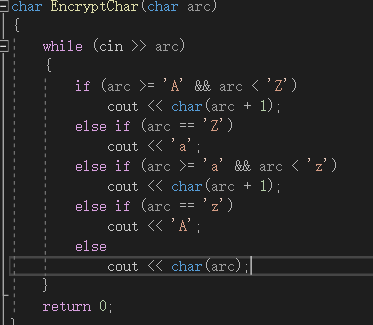
1. 把return 0打成了return 1，(了解return0的原理)还打进了for循环里面
2. 输出一个整数N却用了double，导致最后输出的是科学计数法(补图)
3. 字符没有打引号，直接无法编译。字符串双引号，一个字符单引号
4. 无法输出，改为 cout << (arc + 1)
5. 函数没有返回值，报错

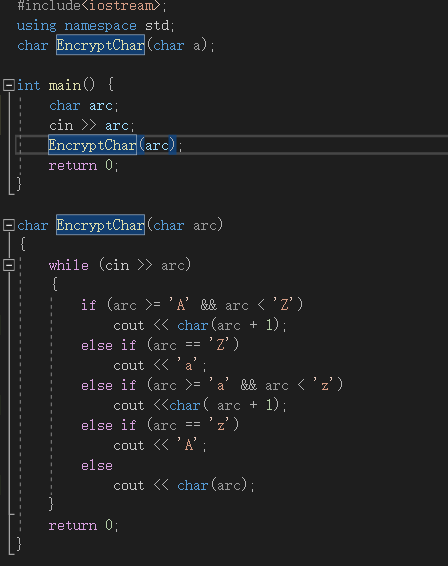
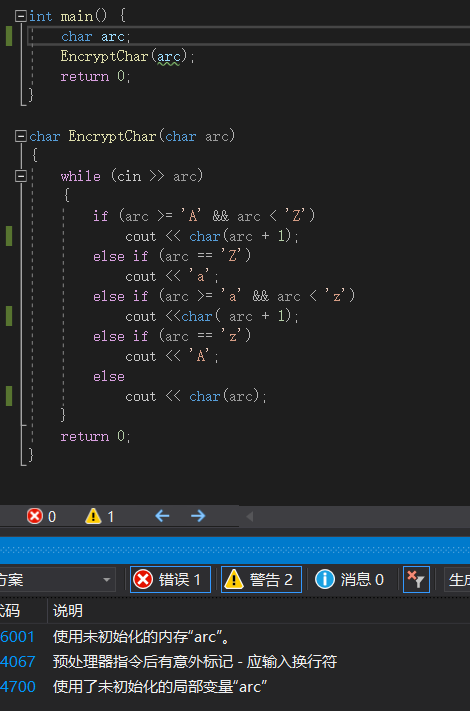


直接敲了个return 0

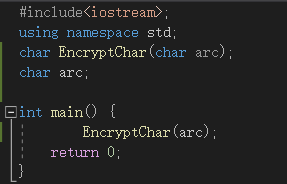
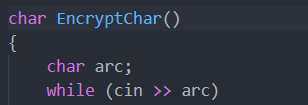
1. 没有搞明白几个if的逻辑

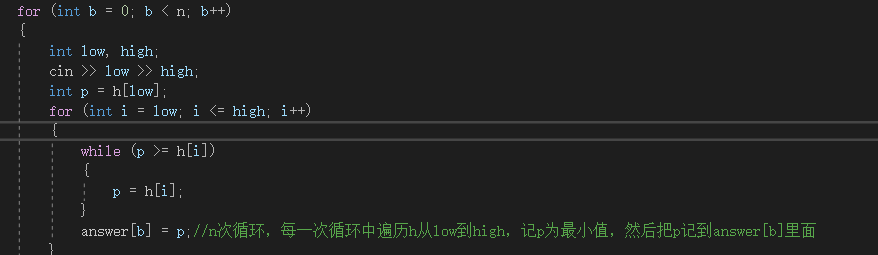


改为

1. =与==
2. 两次cin反而第一次没有输出(了解while cin>='A'的原理)
3. 局部变量未初始化；

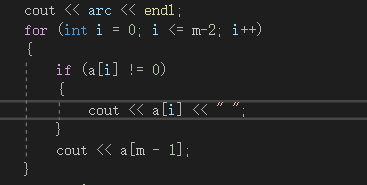
改为全局变量，也不初始化

或者

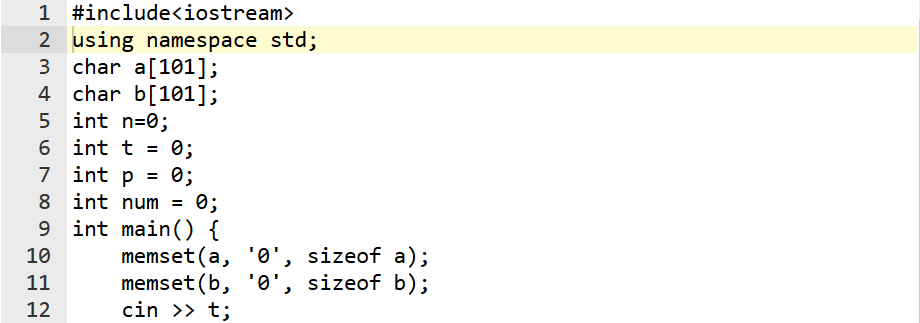
1. 

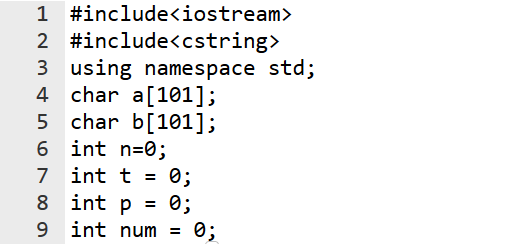
直接死循环，这里while内部没有改p, h, i，如果p>=h[i]成立就会反复地执行while循环体

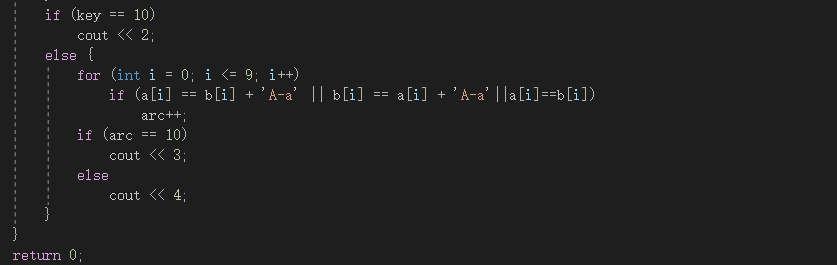
While 改if

1. 把a【m-1】输入了循环节，每次都输出了a【m-1】
2. 输出空格的最后一位单独输出，不加空格
3. 没有include<cstring>导致洛谷上编译不了

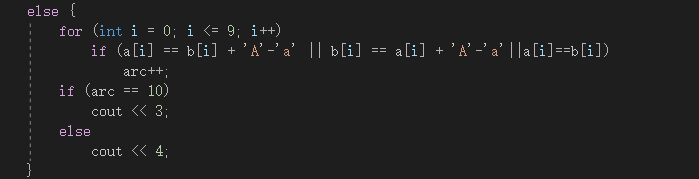
在洛谷和你清OJ上用memset都要cstring





1. 

改为



‘A-a’改为’A’-’a’

1. 没有理清楚left和right的意义

if (a[i] <= a[i + 1] && b[i] >= b[i + 1])

continue;

else if (a[i] <= a[i + 1] && a[i + 1] <= b[i] && b[i] < b[i + 1])

right = b[i + 1];

else if (a[i + 1] > b[i])

{

length =length+ right - left;

left = a[i + 1];

right = b[i + 1];

}

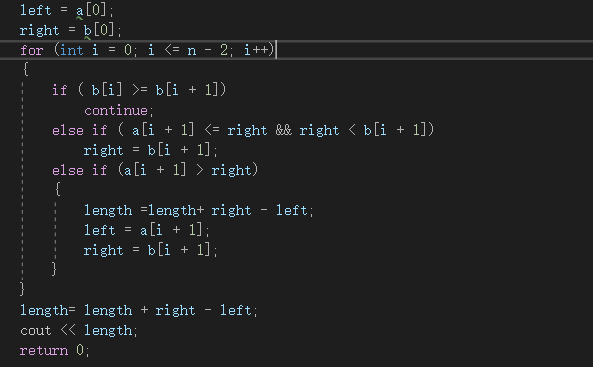
}

length= length + right - left;

cout << length;

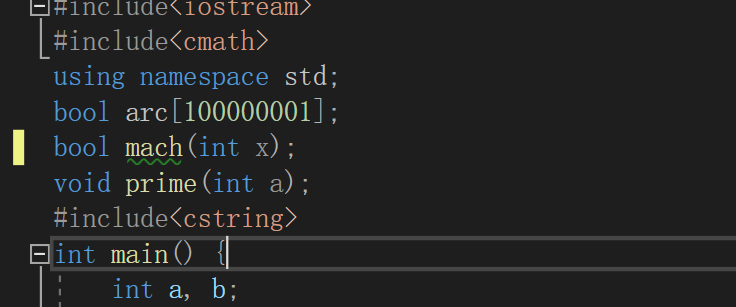
return 0;

}

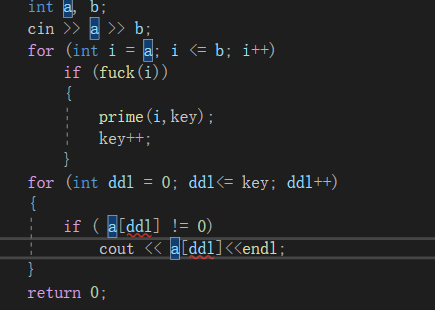


1. 误解了计算机的计算能力，开了一个一个亿级别的数组，直接无法编译。

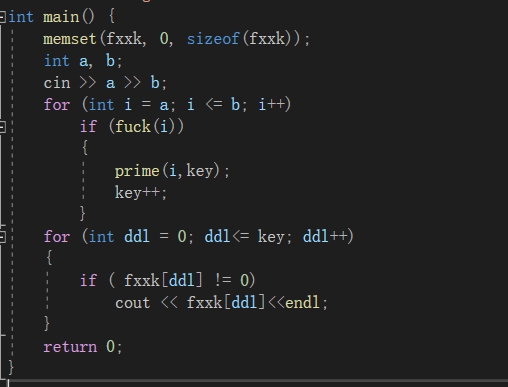
改为先求回文数，再来筛查素数，毕竟回文数比素数少很多



1. 数组命名习惯不好，用了同一个a；导致系统无法识别



改为

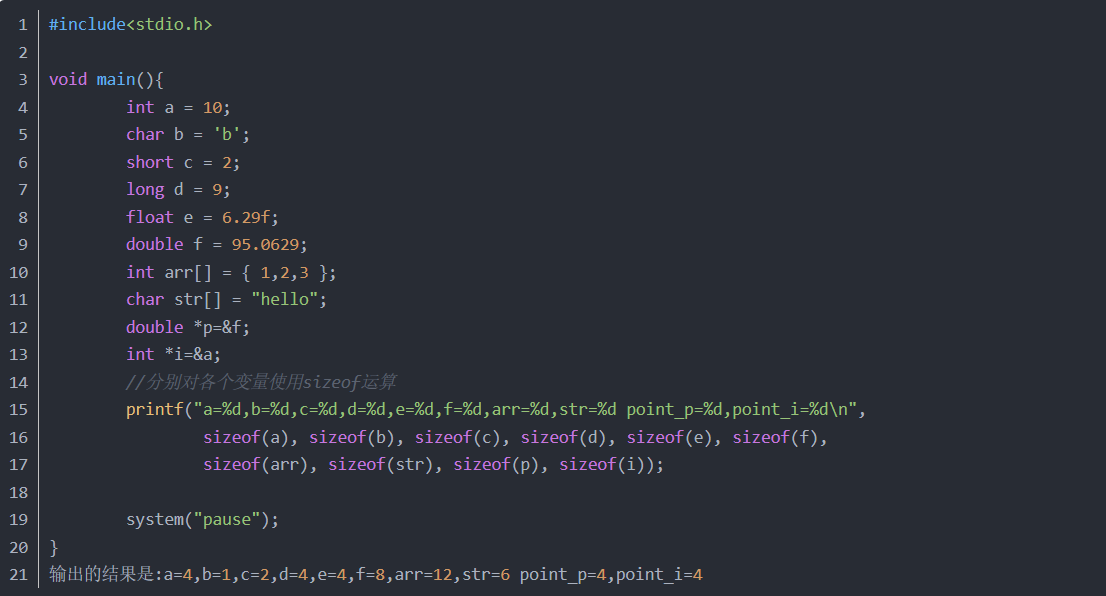


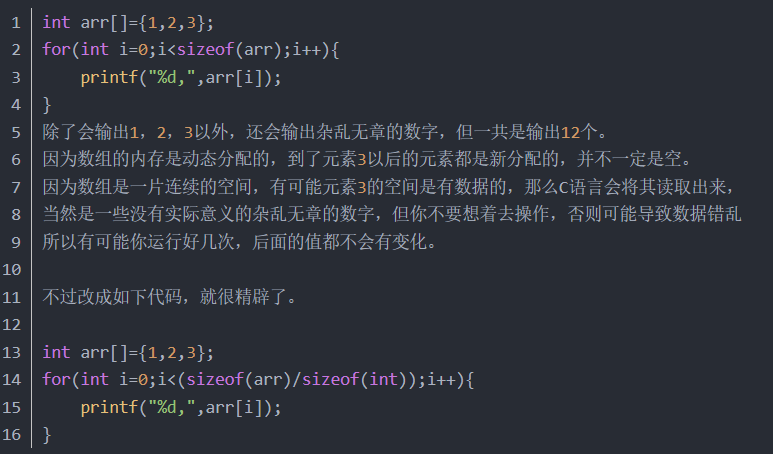
1. （细节）

注意是否把输出行打入了for循环内部

1. #include<iostream>不可以后面加；按住ctrl后点击include有奇效
2. 字符串的长度应该先行给定，比如char a[1000]，但你后面的cin>>a;其实最后一位是有/0的，所以不会读取那么多
3. int n = strlen(a); strlen用于取字符串的长度；

sizeof实际上是获取了数据在内存中所占用的存储空间，以字节为单位来计数。





23.



用动态分配需要指明新数组长度，且最好要delete；

1. Sizeof与memset：

memset（a，0，sizeof（a））；

1. strlen的用法：



1. return 0是个很好的习惯
2. For的写法：for(int i=0;i<10;i++)为什么是<10不是<=9？因为10是实际长度！
3. 数组从0开始其实更加不会错；
4. Debug不必太依赖编辑器自动的标记，但编译器的警告与错误（有错误是进行不了的，警告是系统认为你符合语法的东西不太符合实际编程意图）是必须重视的。
5. Qsort：

Cmpfunc：只有\*（int\*）a里面的int是根据你想排序的类型改变的，其他是规定！

这个写法也有问题（考虑溢出变号）

修改方法：if((\*(int\*)a)<=(\*(int\*)b))return -1;

else If((\*(int\*)a)>(\*(int\*)b))return +1;





与结构体密切相关：排序的大小有自己定义，这就是cmpfunc的作用。

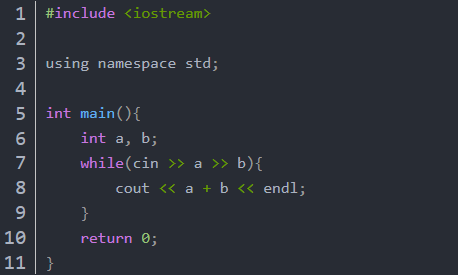
如果a本身是个指针，那么\*a是解引用，就是取出a指向的值。

1. return 0是有明确意义的。Return 大于0和return小于0是有意义的。
2. “指针”无论是全局变量还是动态分配都是可以做加法的。

比如（补图）

1. while（cin>>a）

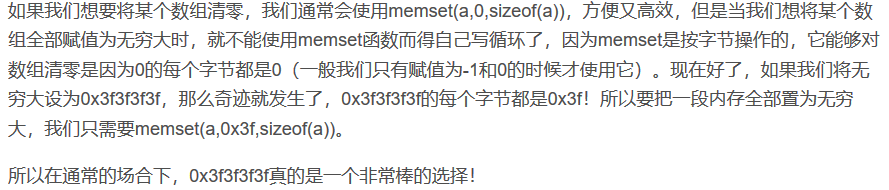
循环用例

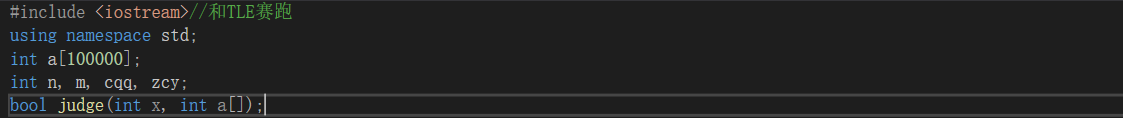


则可以处理多组输入，并且返回多组输出。

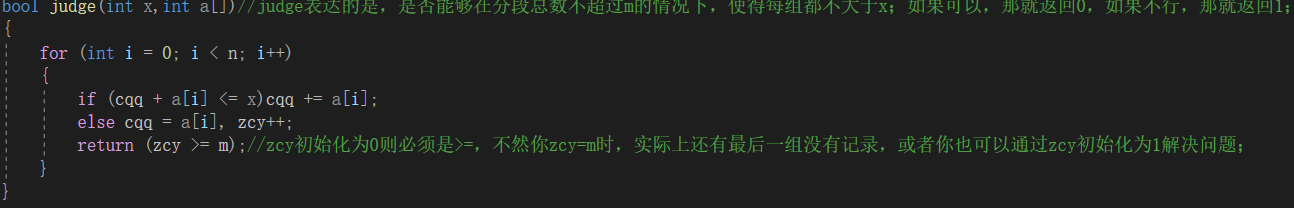
1. 最大上升子串问题：为什么memset（ans，1，sizeof（ans））会导致ans[i]=100691;memset是存字节，你存memset 1就会变成00000001 00000001 00000001 00000001（一个int四个字节，一个字节8位）

Memset的扩展：



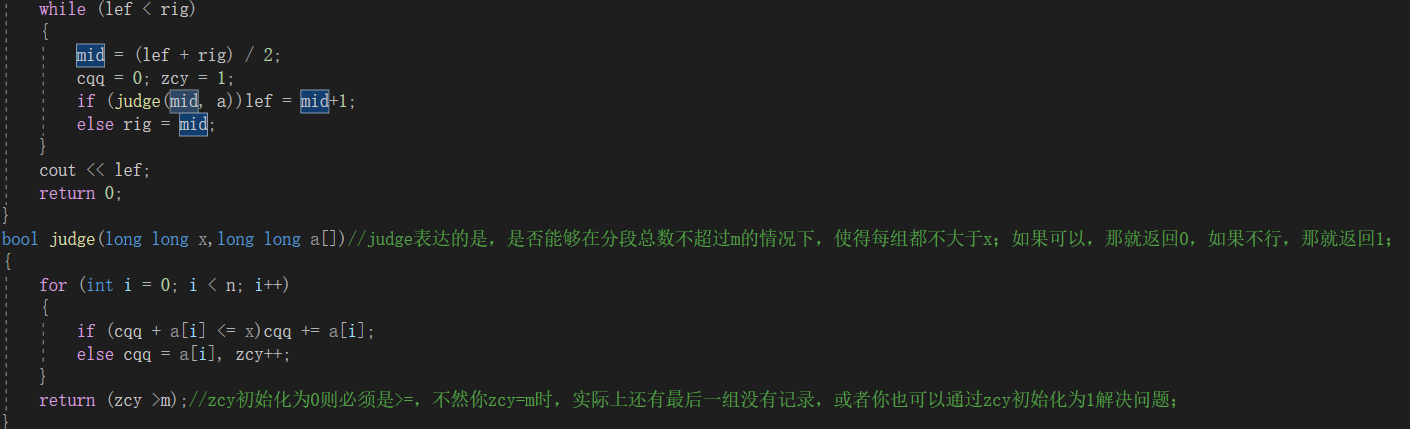
1. 

以此利用全局数组a[];

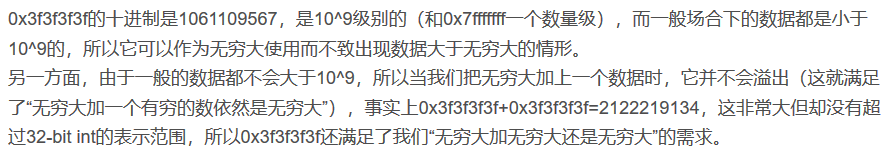
1. 

仔细看，你把return打到哪儿了？这种把关键部分打入for里面的错误经常犯！

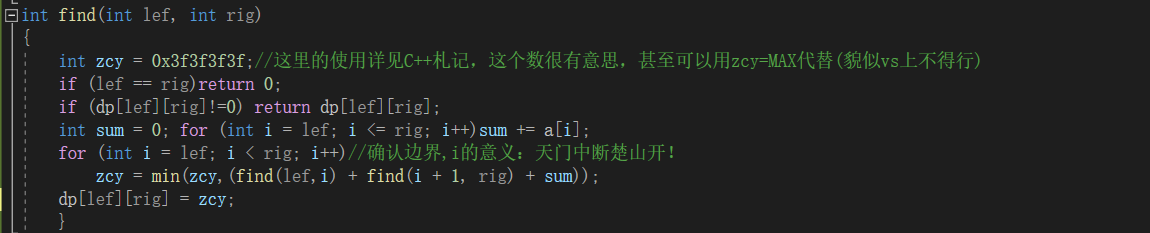
1. 二分法：不能取left和right，推荐l和r；检查几个关键点；不同的二分写法；两边都是闭区间（相对好写）；judge函数的确切定义；mid的取法有讲究，如果l和r都巨大无比，mid会因为加法溢出，用减法或者long long；

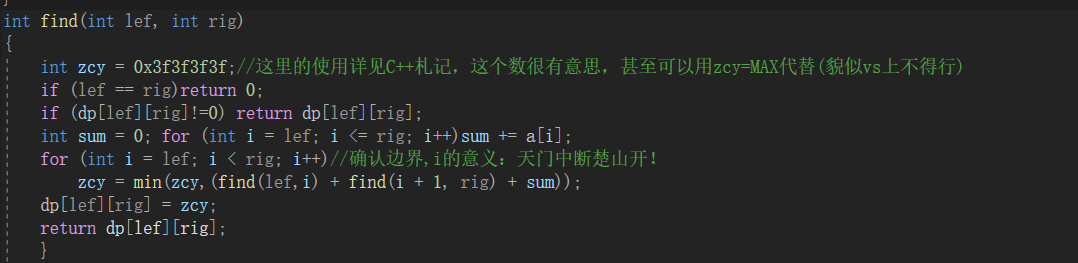


1. int \_min = 0x3f3f3f3f；//int zcy=MAX貌似vs上不得行



1. 写一个没有return的非void函数是很离谱的；





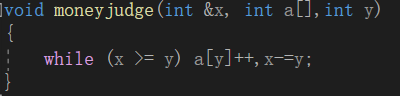
39.养成写{}的好习惯从来不吃亏

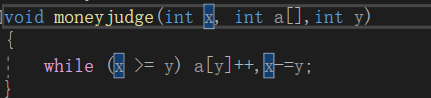
40.ios::sync\_with\_stdio(0);//cin cout加速

41.什么地方可以用“，”？

42.形式参数，实际参数，传引用；

43.函数改变全局变量的值：

传引用：

如果不传引用：z

这样是改变不了x的；

44.一日多错：长多大了还在i=n-1?

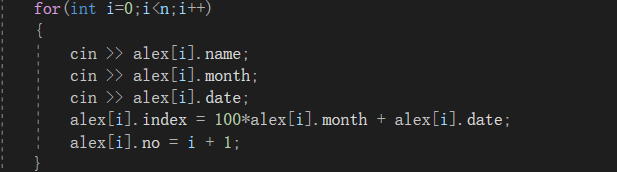


45.不要想着考场先学，用数组代替；一般题目是按照难度排序的，拿下一二题就赢了一半，最后一题只有十分；保二争三；第四题如果一眼就是动态规划，就战略撤退；不要空着，至少输出-1,0；找点边界情况；cin>>n.cout<<n;写一堆if else来骗分；了解链表的基本操作，懂得原理，查询如何用数组模拟链表，建议先学struct，可以用sort，可以自定义排序方法；用不到stl，只用学会sort，最好直接用qsort；

46.全局变量是有默认为0的；局部变量没有；

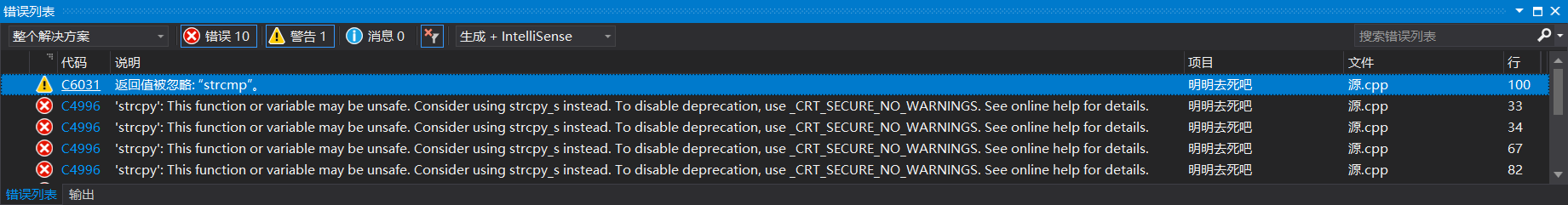
47.结构体快排&&稳定性处理：





48.vs有的时候会爆出奇怪的错误

比如说这种：



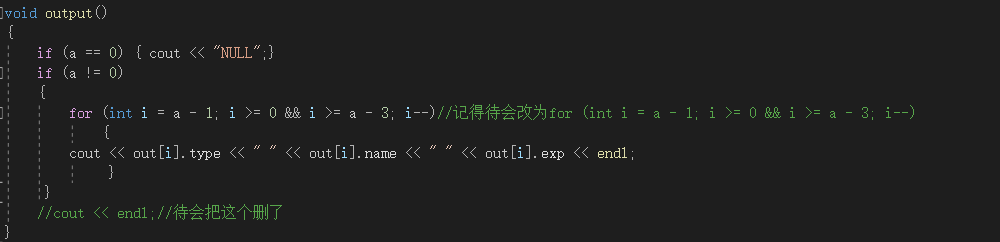
这时候顶头加上

#define \_CRT\_SECURE\_NO\_WARNINGS

如果OJ报错，那就交的时候再删去

49.主函数以外的函数相互调用

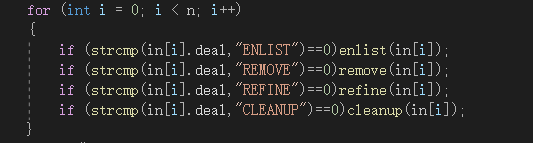
要么就写出次序来，被调用的写在前面，要么就先声明了在主函数后面调用

50.定义void函数

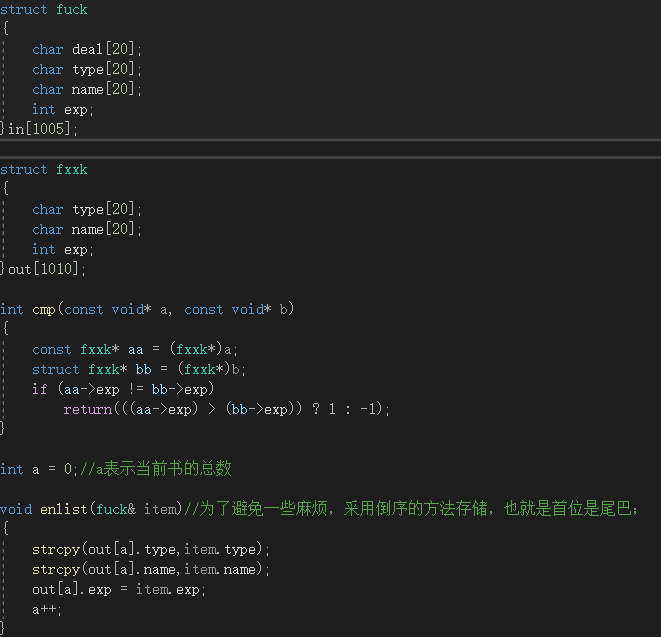
好歹也要加括号，而且使用的时候也要加



51.把结构体作为参数传入函数



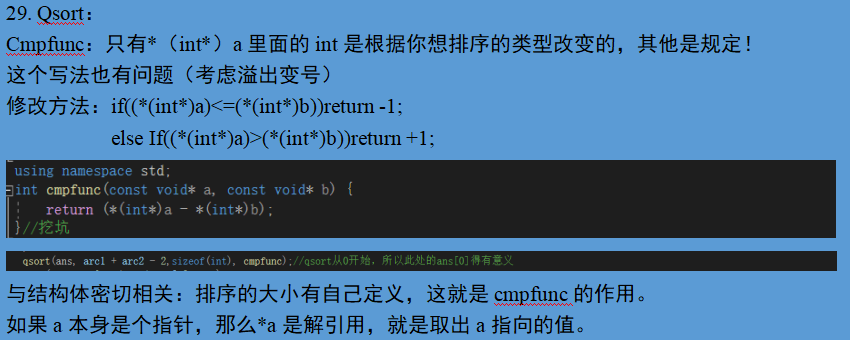
而函数定义部分写法：



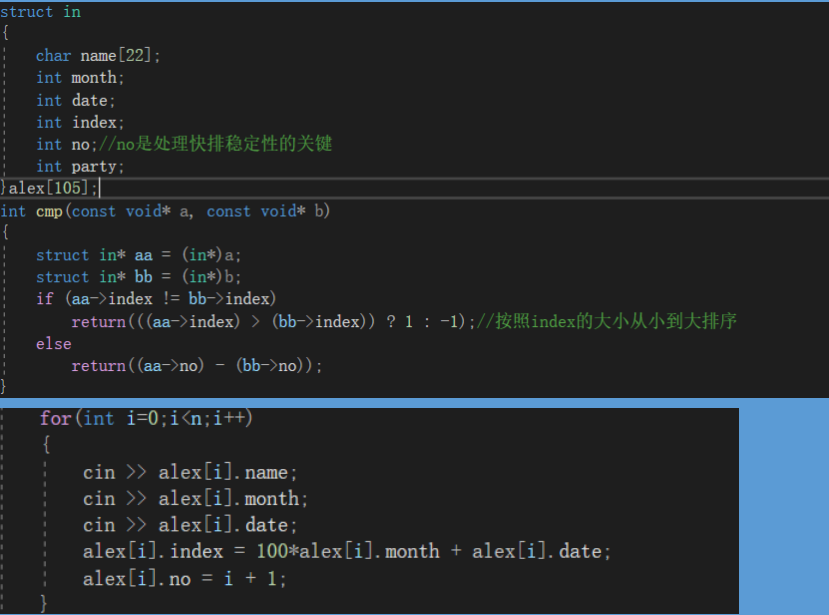
注意到传了引用，且是fuck类型的item（即为fuck&item）

52.快排小结

A.最基本的数组快排

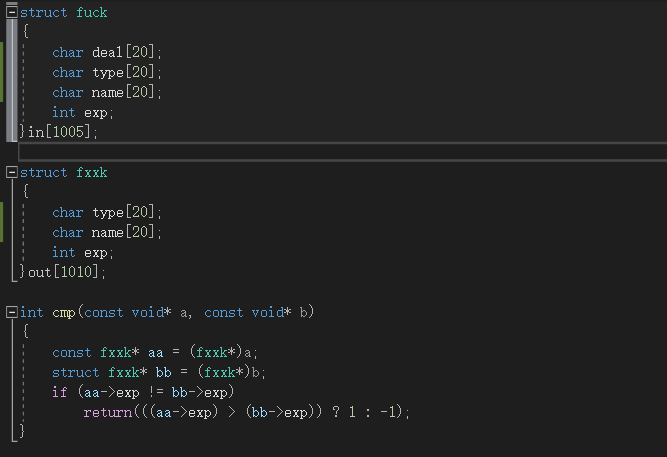


B.结构体快排&&稳定性处理：



注意到你的cmp里面，in是struct的类型，而不是某种struct的名字

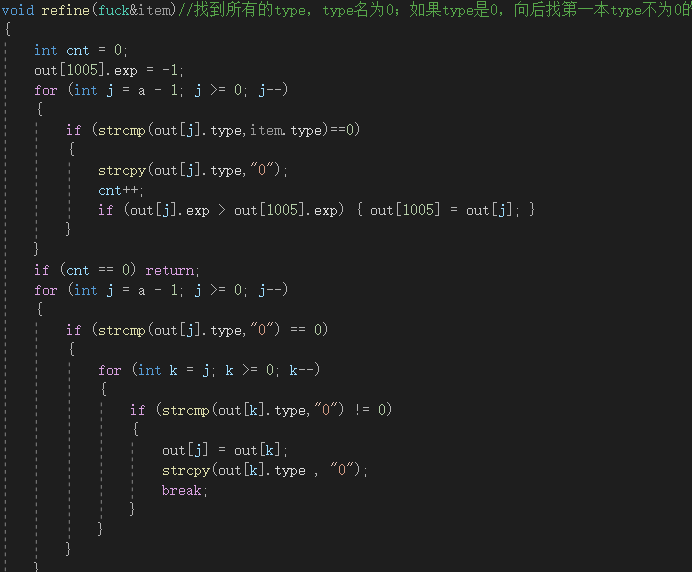
比如这里



C.带字符的结构体qsort

不要用string，老老实实用char组

问题是，char的比较和赋值是这样的：



极其容易将strcpy和strcmp搞混，甚至直接用字符串相等。字符串不能直接赋值，但是放在结构体里，可以通过结构体赋值而实现对字符串赋值；