**数据结构实验报告**

**（四）**

**学院 自动化学院**

**学号 08022311**

**姓名 陈鲲龙**

**日期 2024-1-3**

**实验内容**

现在社会上各种排行评比越来越多，其中较有影响的就是十大流行歌手排行榜。为了评比、统计和发布十大流行歌手排行榜，需要能够完成以下功能的软件：

①统计和显示排行榜；

②为喜欢的歌手投票；

③查看上榜歌手的主打歌曲；

④查看歌手的票数。

【问题分析】

通过对一段时间的投票情况进行统计，按得票数取前十名，即可得到十大流行歌手排行榜。

由于需要对一段时间的投票数进行统计，因此必须保存投票情况。一个比较好的选择是使用文件保存，可以把每一个月的投票情况分别保存在一个文件中，这样一年的投票情况只需要12个文件即可。为了节省篇幅，可只使用一个文件。

为了查找的方便，对文件中的数据按哈希表形式组织，以歌手的名字作为关键字。为歌手投票，就是在哈希表中查找歌手的过程。若找到歌手，则为歌手添加一票；若没找到歌手，则在哈希表中插入该歌手。查看歌手的主打歌曲和查看歌手的票数，都是对哈希表进行查找操作。

为了完成统计和显示排行榜，可以二叉排序树，因为二叉排序树的中序遍历序列就是一个有序序列。当然，也可以使用各种内排序方法。

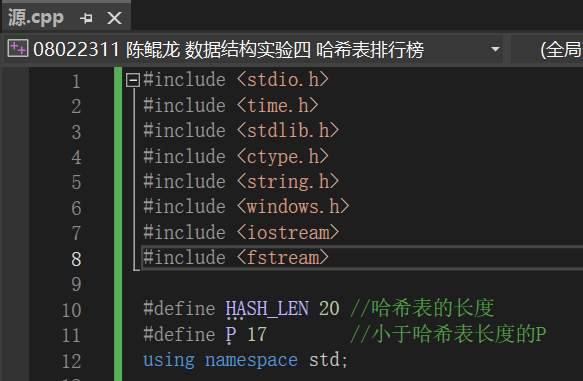
1. **需求分析**

**设计任务：**本实验的需求是实现一个歌手排行榜程序，查找和存储使用哈希表实现，排序使用中序遍历二叉排序树实现

**输出形式：**基于cout打印程序对应内容

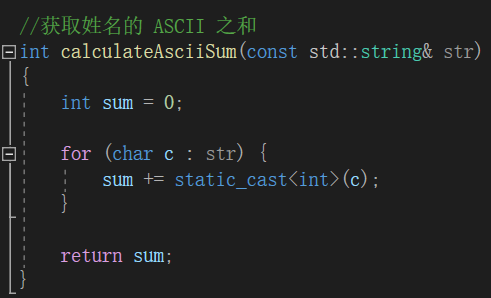
**测试：**先根据程序窗口的提示输入对应数字选项即可。

1. **设计思路**

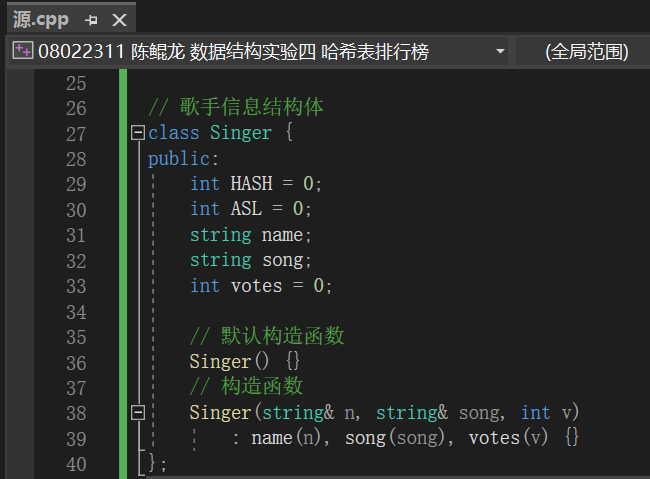


首先是宏定义哈希表的长度为20，而哈希函数使用key mod P，关键字是歌手名字的字母ASCII码之和，P取小于哈希表长度的17。

把string变为char然后累加计算出ASCII码之和作为哈希表的关键字



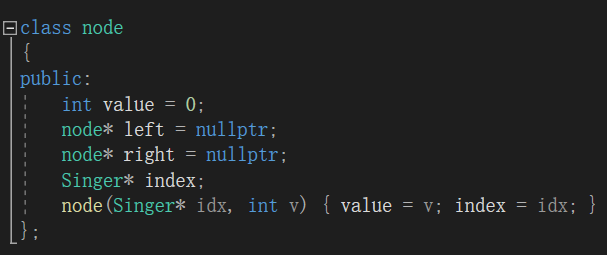
然后定义歌手信息的类，包含歌手的名字name、歌曲名字song、票数votes、在哈希表中的序号（即位置）HASH、查找正确数ASL。



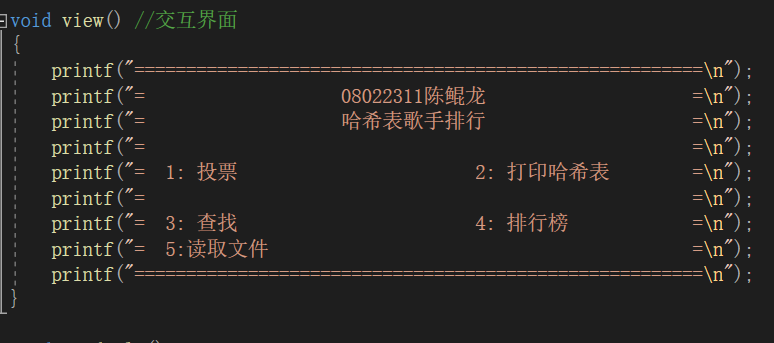
然后哈希表就存储在以下这个元素类型为指向Singer类的指针的数组，初始化为空（学到了初始化自定义的类要加大括号{}）



再定义二叉排序树的结点



交互界面：



12345在主函数中还是用switch case语句实现

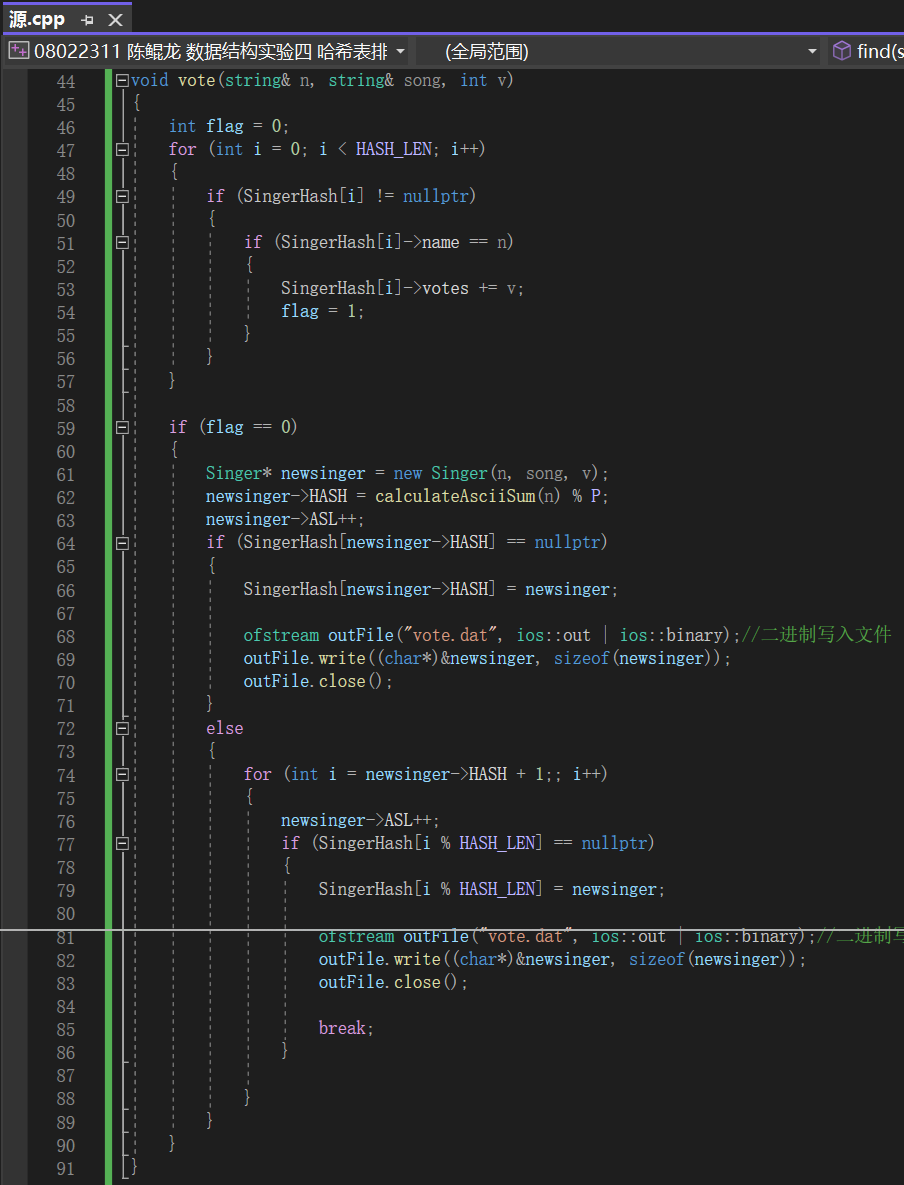
首先看1：投票

实现思路如题干所说，在哈希表中查找，找到就加票数，没找到就在哈希表中加入这个歌手。

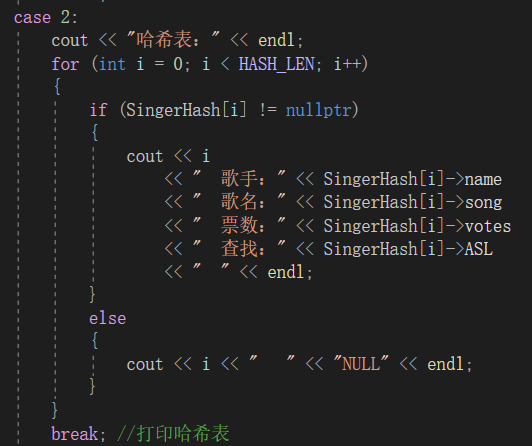


看具体实现函数vote()

现根据name来找，找到就修改票数vote+=v，没找到就new一个指向新建singer实例的指针，然后就是在哈希表中开放寻址找空位，如果它对应的key位置有空那放入数组就结束，如果哈希表发生冲突，那就逐个往后找空位。

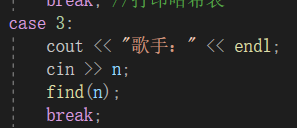


然后看2打印哈希表：

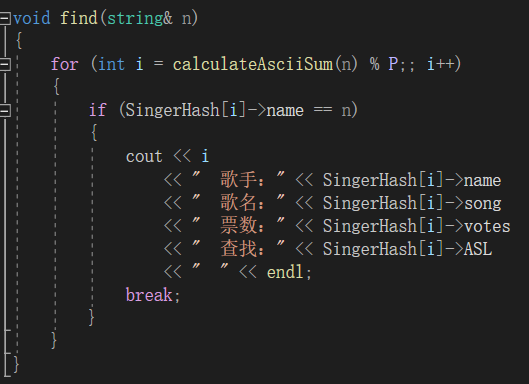


就是遍历打印SingerHash数组每个元素Singer指针指向的Singer实例的内容

然后看3查找

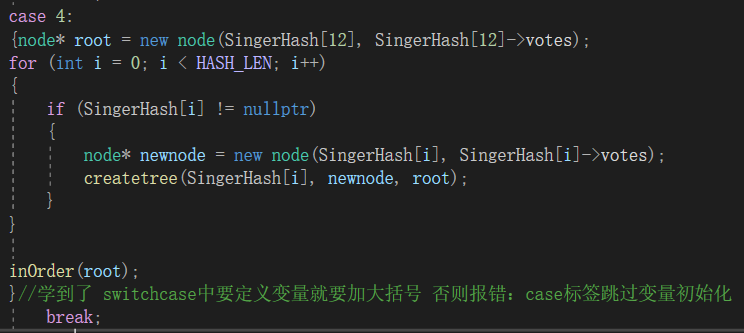
、

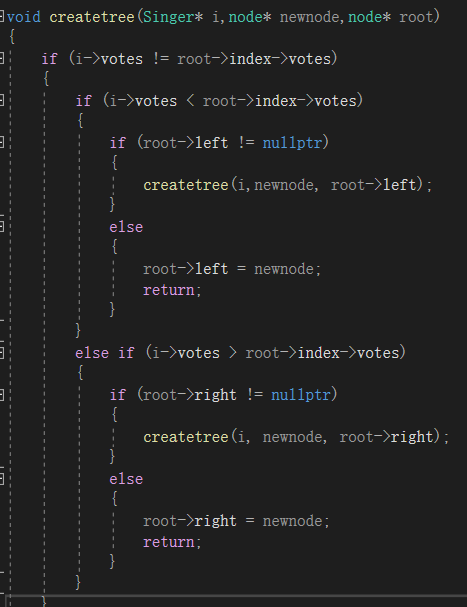
具体find函数的实现，实现逻辑很简单，就是依据name找到并打印



然后是4排行榜，这里涉及到二叉排序树的构建和中序遍历：

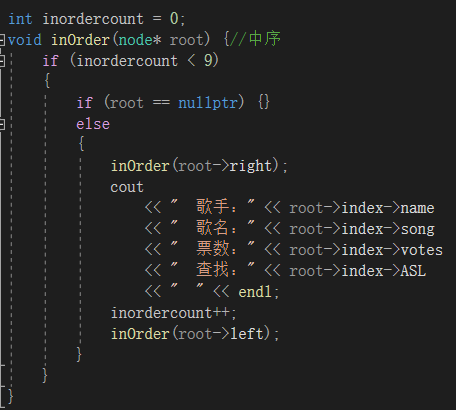
先创建一个根节点，哦，我写到这里才想起来，这里有个小缺陷，12是因为测试时我总是输入a，它的ASCII为97，%17=12，就是说这个根节点不能建立于空的还未实例化的singer上，必须指向一个已经有的singer实例的node结点上；然后就是遍历整个数组，将非空的结点比大小填入二叉排序树，比大小是需要递归的，就是createtree这个函数。





每次新进来一个数，就逐层与根节点比大小，直到比到最底层，那就指针指它，即添加操作。

中序遍历和之前某次作业是一样的，两个小改动，排行榜只有前十，所以加了个inordercount用来计10个数，又为了从大到小输出，把中序遍历左中右改成了右中左，这样就从递增变成递减了。



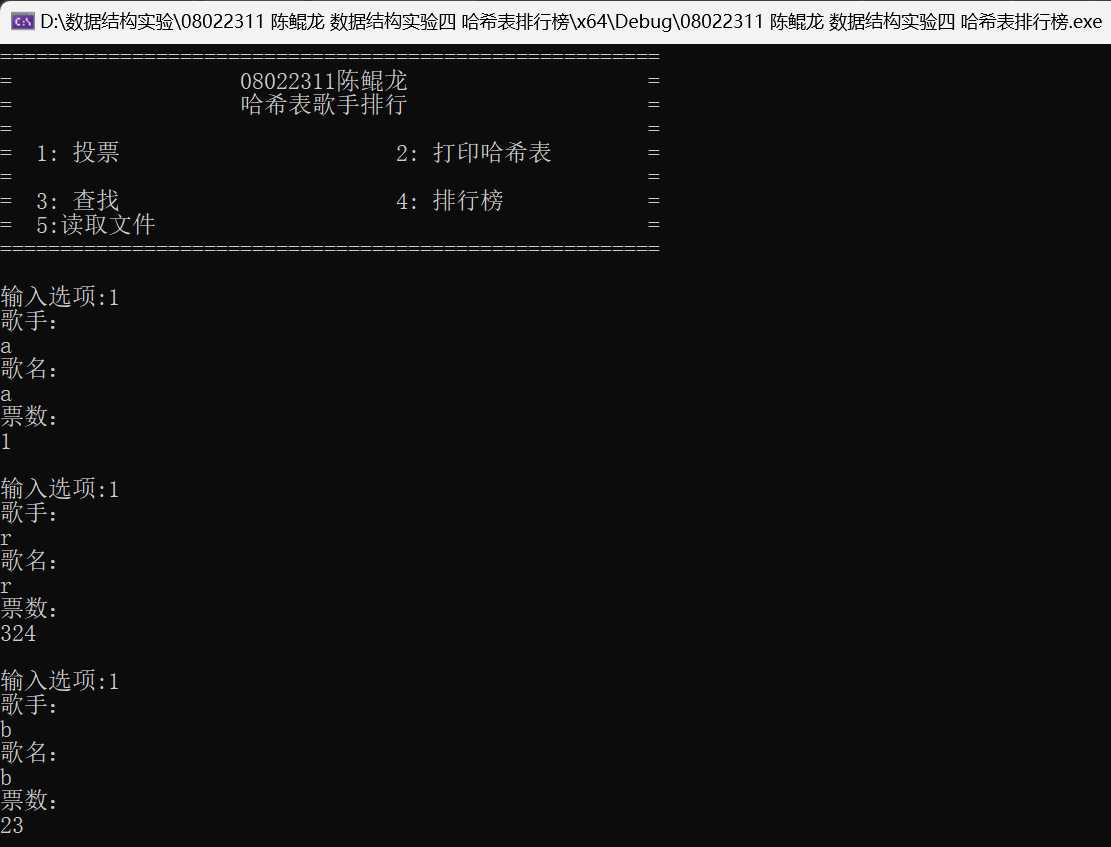
最后写入、读取文件的部分，我从上学年的书上找并写在程序中，我要写入的是哈希表，而且元素为自定义的singer类，所以不能是txt文件，而是二进制文件，但测试不太成功，无法保存。

1. **调试分析**

&

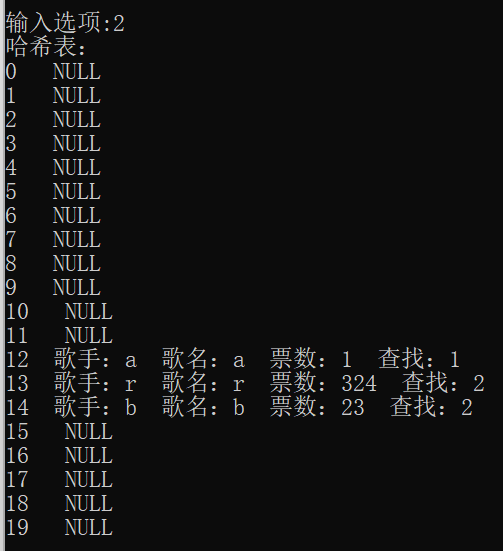
1. **测试结果**

调试与测试：

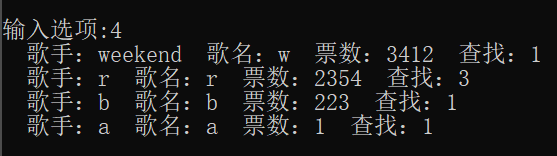


我们先输入一些内容

然后选项2查看哈希表



可以看到r和a都应该是12，而这里开放寻址解决了这个冲突，同理b应该是13，也是往后顺延一位。



选项四的排序功能也正常。

1. **实验总结**

这次实验加强了对于哈希表的理解，包括对于字符串的哈希函数的建立、冲突的解决等问题，也理解了二叉排序树的相关内容，包括递归函数的处理更熟练了。

1. **附录**

本实验代码仅 源.cpp