**Project 5实验报告**

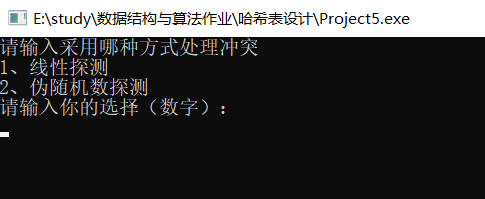
20337263 俞泽斌

1. 程序功能简要说明。

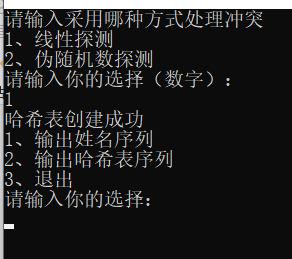
这个程序首先就是读取一个文本文件，也就是data.txt里的数据，就是30个人名的拼音，然后通过建立哈希表的方式，最后将建立的哈希表输出

对于每个人名的关键字，这里采用的是按字数取ASCII码然后相加的处理方式，然后用最大数来进行除留余数法构造，在遇到冲突的情况下采用伪随机序列探测和线性探测来进行判断，哈希表中对于每一个的查找都保存了他们的查找长度，用作具体的效率判断。

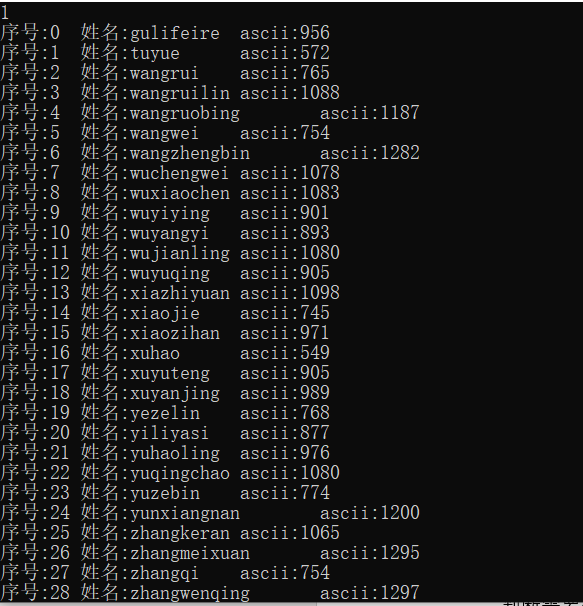
1. 程序运行截图，包括计算功能演示、部分实际运行结果展示、命令行或交互式界面效果等。



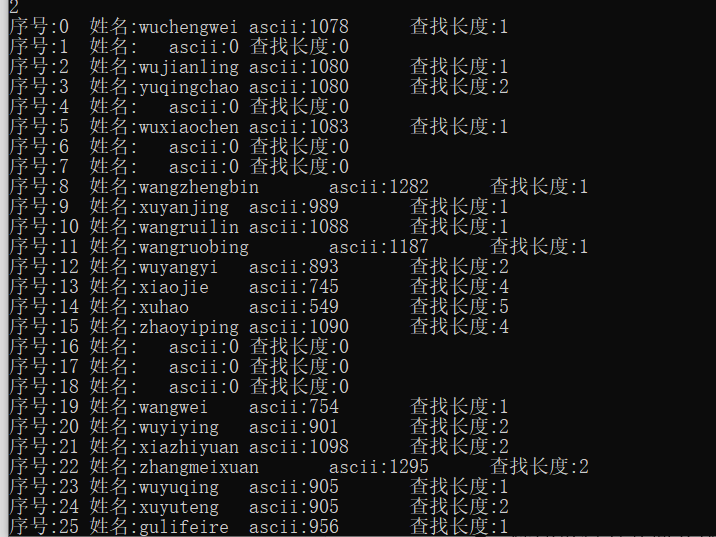
首先这是开始的界面，因为我做了两个处理冲突的方式，一个是线性的方式，也就是每次遇到冲突后加一，另一个是伪随机数的探测模式，也就是遇到冲突后加上一个随机数，这里的随机数我用的是在-key到max-key的来保证最后得到的数一定在整个哈希表序列范围内，不至于发生溢出错误。

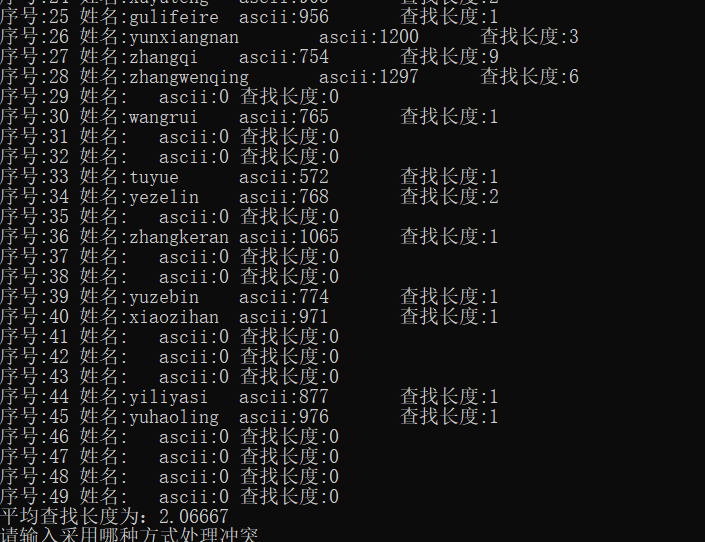


判断需要输出什么序列，姓名序列主要是用来看姓名，序号以及对应的ASCII码值，防止在读入的时候就出现问题。



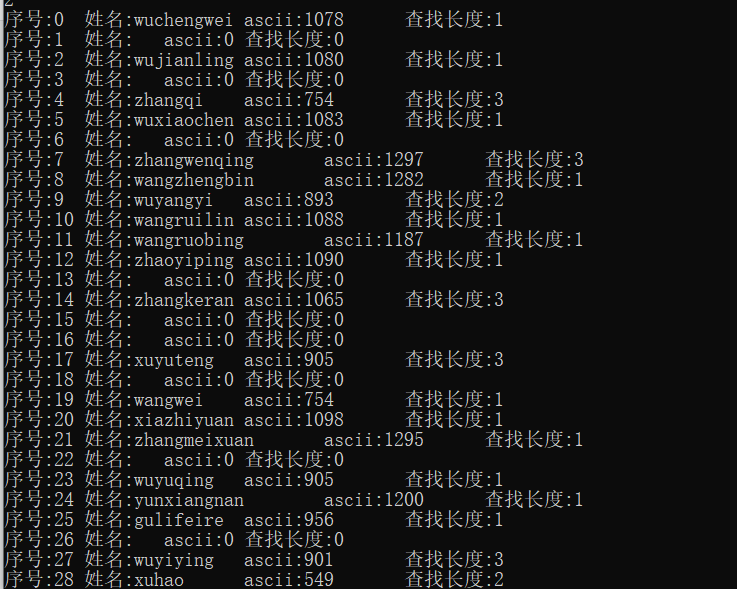
这里就是对应的姓名及ASCII码表格

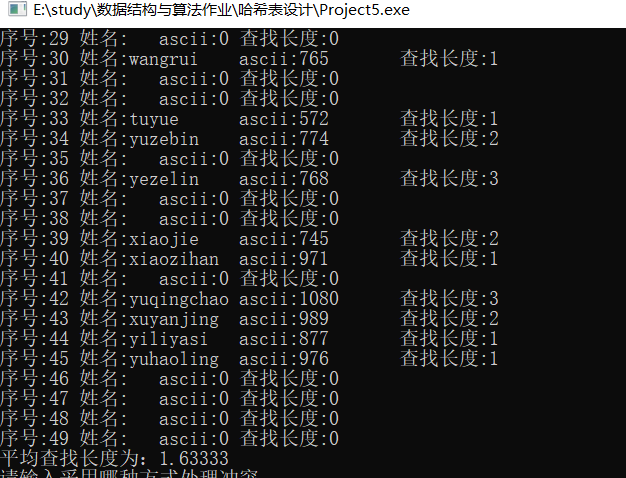




这一次采用的是线性探测的方式。

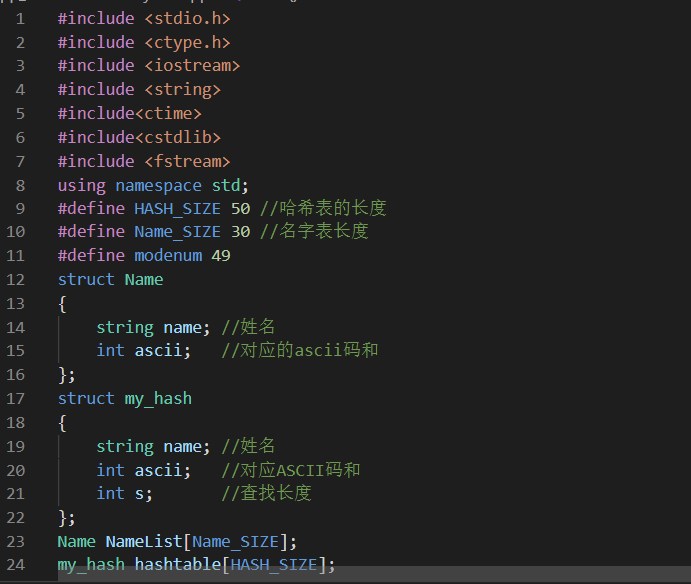
这里就是我们所有的哈希表及上面有的信息，如果哈希表中有的表明数据以及存储进去了，并且我们对于哈希表中的每一个数据都输出了他的查找长度，也就是通过对应的哈希算法需要经历的时间，最后输出了他的平均查找长度。



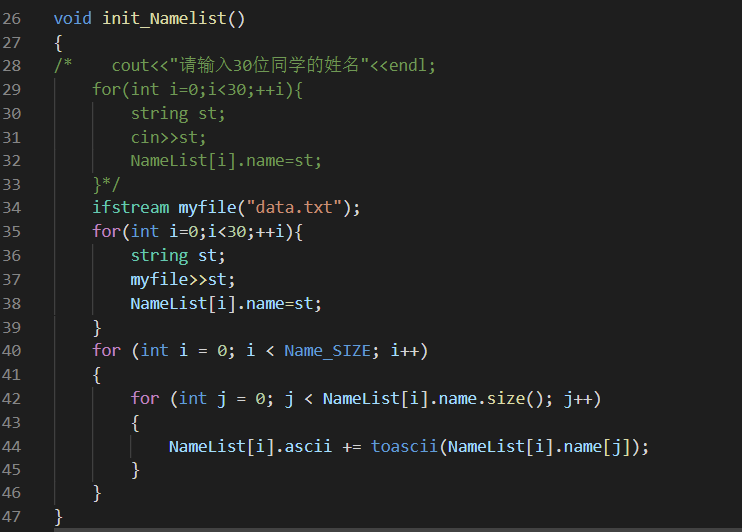


这里采用的方法是伪随机序列的探测方法，可以看到平均的查找长度在1.633，比起上面的探测就少了许多，但是伪随机毕竟还是随机，平均查找长度的波动性还是存在的，不过都是在2以下，也算是完成了任务。

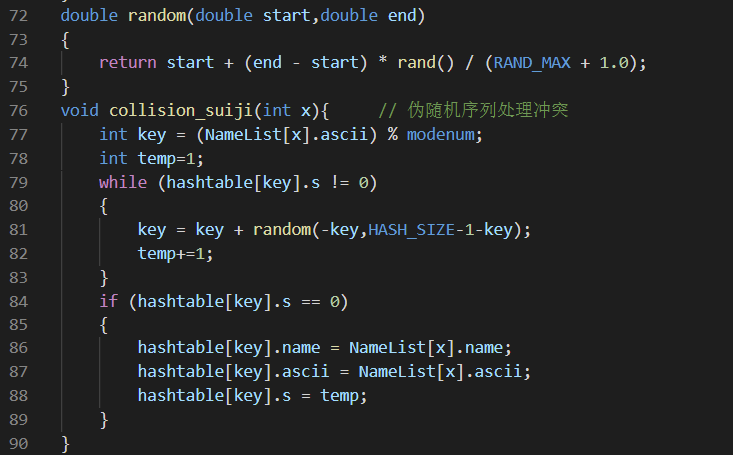
1. 部分关键代码及其说明。



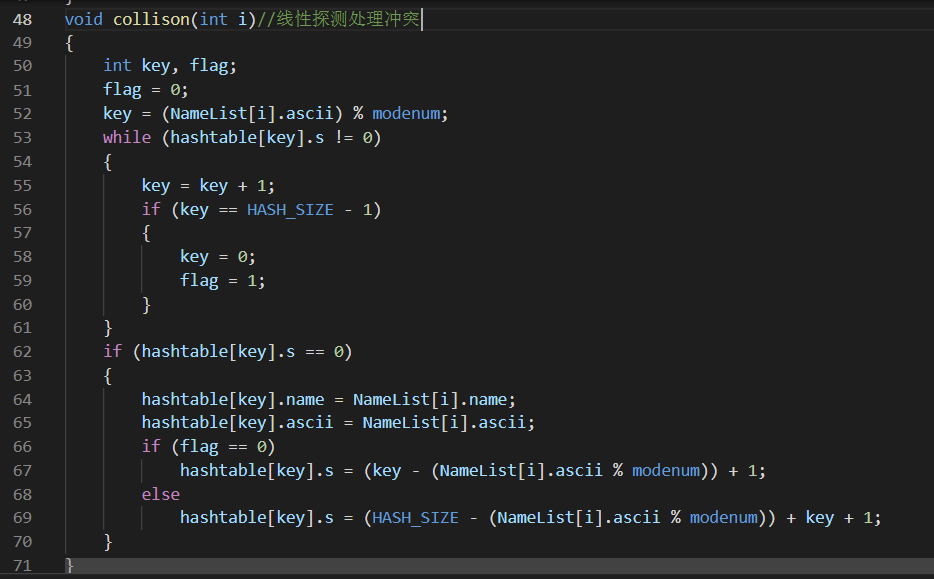
首先是两个结构体的定义了，一个是姓名列，一个是hash算法所产生的结构。



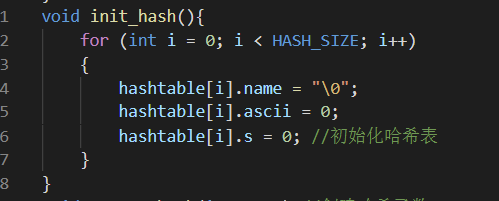
读入数据的部分，这里采用的是从文件中读取的操作，读取的是data.txt里的文件，用字符串的形式存储，然后通过逐字逐句来+每个字的ASCII码来进行对于姓名表里ASCII码的更新。



这里是伪随机序列探测法的操作了，首先我们用的是一个rand函数，然后通过数学的公式来得到一个random函数用来进行一个范围内的随机数，然后如果除留余数法得到的key冲突的话，就采用加上一个random函数，在-key到max-1-key的方式让得到的key永远在哈希表内。并更新查找长度。

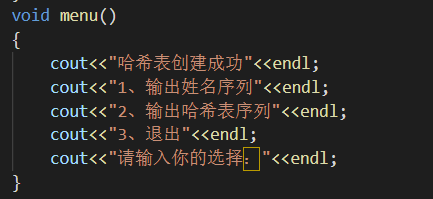


这里是关于线性探测的方式了，其实主要是通过除留余数法来找到key，然后如果key的位置上不是0，也就是有数据了，我们就用一个循环加一直到找到一个空位置，注意如果到了哈希表末尾要从头重新开始，并且在查找长度中加上关于前面一个哈希表里的长度了。





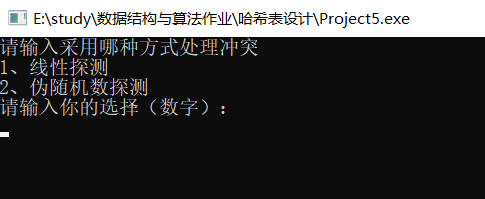
这是一个主体的部分了，首先将哈希表初始化，然后通过除留余数法得到key也就是位置，如果没有数据就放入数据，如果有数据就采用处理冲突的方法，这里的type就是写的两种处理冲突的方法，包括伪随机序列和线性的方法。



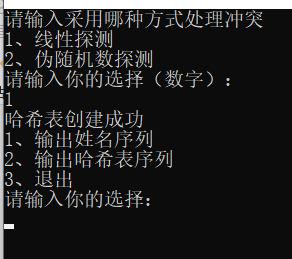


这里主要就是前端的设计来方便操作了。

1. 程序运行方式简要说明。



这个程序首先是输入要用哪种方式来进行序列的探测，也就是处理冲突的方法，具体到代码里就是输入的type，用在create\_hash里来判断探测序列



之后就是具体要输出什么了，因为输入的数据都保存在了data.txt中，这里采用从文件中读取的形式来方便操作，然后想看namelist有没有读错的选1

想看具体的哈希表的实现，也就是哈希表每个位置上有没有数据，有什么数据的选2，里面还有具体的平均查找长度。

