石 家 庄 铁 道 大 学

**实 验 报 告**

**实习五 哈希表应用**

题 目： 哈希表设计

班 级： 信1901 –4

姓 名： 闫竞存

学 号：20194127

日 期：2020.11.03

**1. 实验题目**

哈希表设计

[问题描述]

针对某个集体中人名设计一个哈希表，使得平均查找长度不超过 R，并完成相应的建表和查表程序。

[基本要求]

假设人名为中国人姓名的汉语拼音形式。待填入哈希表的人名共有 30 个，取平均查找长度的上限为 2。哈希函数用除留余数法构造，用线性探测再散列法或链地址法处理冲突。

[测试数据]

取读者周围较熟悉的 30 个人名。

**2. 需求分析**

设计一个哈希表存储30个人名，使用链地址法处理冲突，使其平均查找长度不超过2

**3. 概要设计**

本程序的类、全局变量和函数及其主要作用:

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 作用 |
| ListNode | 类 | 哈希表节点类 |
| names[] | 全局变量 | 姓名 |
| hashTable[] | 全局变量 | 哈希表 |
| readFile | 函数 | 读取文件 |
| enterName | 函数 | 输入姓名 |
| insert | 函数 | 向哈希表中插入数据 |
| hashCode(string) | 函数 | 计算哈希码 |
| findInHashTable(string) | 函数 | 在哈希表中进行查找 |
| main(null) | 函数 | 进行相关操作的选择 |

**4. 关键代码的详细设计**

哈希表的节点设计

// 哈希表节点

class ListNode {

public:

    ListNode() {

        this->data = "";

        this->next = NULL;

    }

    ListNode(string data, ListNode\* next) {

        this->data = data;

        this->next = next;

    }

    ListNode\* next;

    string data;

};

向哈希表中插入数据

// 向哈希表中插入数据

// 在此之前，已经将姓名存到了string数组names中

void insert(ListNode\* hashtable, int posi, string data) {

    // hashtable 是哈希表， posi是要插入的位置，data是要插入的数据

    if (hashtable[posi].next == NULL) {

        hashtable[posi].next = new ListNode(data, NULL);

    } else {

        ListNode\* temp = hashtable[posi].next;

        while (temp->next) temp = temp->next;

        temp->next = new  ListNode(data, NULL);

    }

}

在哈希表中查找相应的值

//在哈希表中查找指定字符串，成功返回1，否则返回0

int findInHashTable(string nameToFind) {

    ListNode\* temp = nullptr;

    if (!(temp = hashTable[hashCode(nameToFind)].next)) return 0;

    while (temp) {

        if (temp->data == nameToFind) {

            return 1;

        }

        temp = temp->next;

    }

    return 0;

}

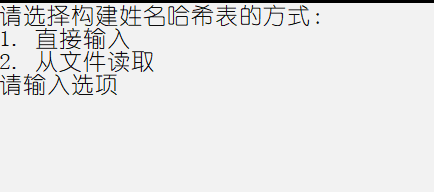
**5. 遇到的问题以及解决方案**

我感觉我的问题主要出在函数的功能划分不清晰，主函数中加入的比较多的和流程相关的的业务逻辑。程序耦合度高。

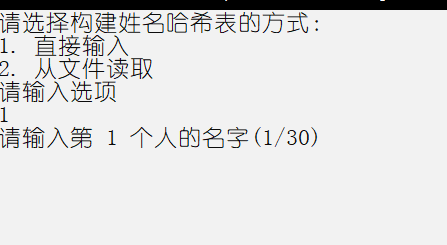
其他方面没什么问题了。

**6. 用户使用说明**

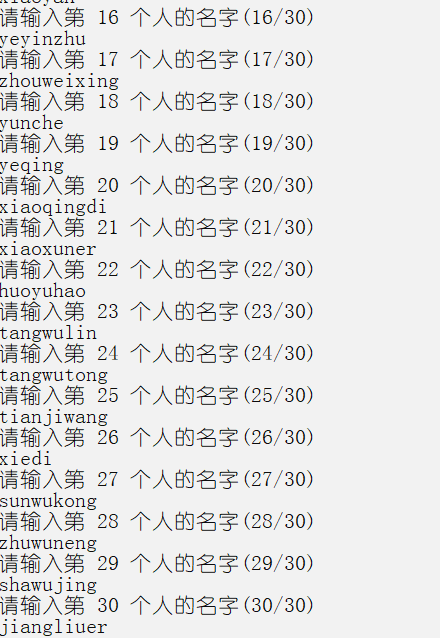
打开软件



输入选项



依次输入30个人的人名



然后弹出提示：数据插入成功

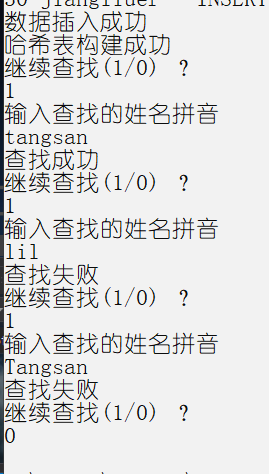


并询问是否继续查找操作：

在这里输入0或者1代表继续操作或者退出程序

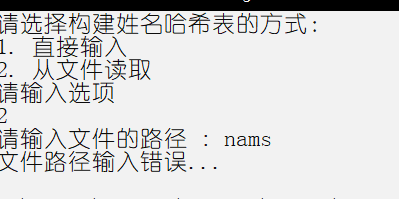
注意：这个查找是完全匹配查找，而不是模糊查询，只有大小写完全一直才能查找成功。

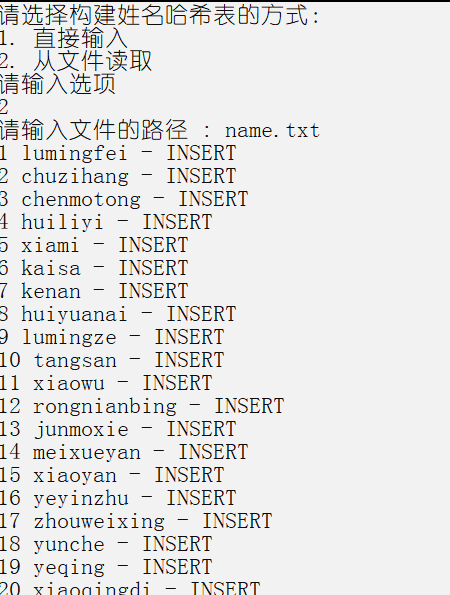
否则查找失败。



也可以直接将姓名输入到文件中，从文件中读取姓名

此时应该输入文件的完整路径或者文件相对于这个文件的相对路径





此时会读取前30个数据

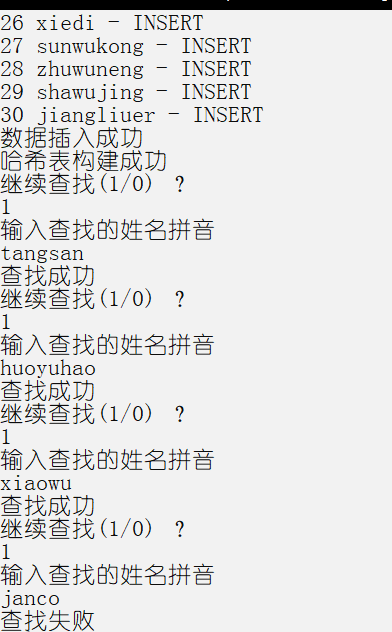
剩余功能已经测试过了，故不再重新测试。

**7. 测试结果**

测试数据:

测试数据一共30个数据，文件中也是这些数据

|  |
| --- |
| lumingfei  chuzihang  chenmotong  huiliyi  xiami  kaisa  kenan  huiyuanai  lumingze  tangsan  xiaowu  rongnianbing  junmoxie  meixueyan  xiaoyan  yeyinzhu  zhouweixing  yunche  yeqing  xiaoqingdi  xiaoxuner  huoyuhao  tangwulin  tangwutong  tianjiwang  xiedi  sunwukong  zhuwuneng  shawujing  jiangliuer |



具体查找方式见“用户使用说明”