**电子科技大学**

**计算机科学与工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 软件开发综合实验**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

# 实 验 报 告

**学生姓名： 学 号： 指导教师：**

**实验地点： 实验时间：**

**一、实验室名称：**

**二、实验项目名称：电话号码本**

**三、实验学时：8学时**

**四、实验原理：**

1.将终端输入的人名（string）转化为关键字（key）。

2.利用hash函数将key转化为location（hash地址）。

3.利用hash地址读取查找表，解决地址冲突问题，并进行关键字匹配。

**五、实验目的：**

1.熟悉基本数据结构（数组、链表、hash表）的使用。

2.掌握<string>、<vector>、<map>等STL容器。

3.了解几种数据结构及多种查找算法，并掌握hash查找算法。

4.学会利用数据结构和算法编写简单程序。

**六、实验内容：**

.提供一个动态电话号码查询系统，可以通过2种方式（联系人姓名和电话）进行电话信息查询，查询的信息至少包括三部分（姓名，电话，地址）。当采用联系人姓名进行查询时，如果查有此人，则显示查询信息，并提示用户是否修改该联系人信息；否则，提示用户是否添加该联系人到系统。当采用电话进行查询时，允许多条信息，即电话号码可重复。

1. **实验步骤：**
2. 定义联系人结构体以及顺序表结构体，定义联系人结构是为了方便统一管理一个元素拥有的多个信息，比如姓名，电话，地址等。并且由于需要用到哈希查找，故需使用顺序表的数据结构，部分代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct Contact //定义结构体联系人 包含元素：姓名 电话 地址（为字符串类型） 标志变量int型表示该节点是否已经存了人了

{

char name[20];

char telephone[15];

char address[20];

int flag; //标志变量 =0或者未赋值时表示此处无人 =1表示已经有人存在这里了

}Contact;

typedef struct Sqlist //定义结构体顺序表 包含：Contact型指针用于指向第一个 nn表示现在已有联系人 mm表示电话本最大容量

{

Contact \* elem;

int nn;

int mm;

}Sqlist;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 下面对于这两个结构体的相关操作，以实现功能，部分函数头如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void InitialContact(int size, Sqlist \* telephonehash, Sqlist \* namehash) //联系人初始化操作

void TelephoneInfo(Sqlist telephonehash, Sqlist namehash)

//显示当前电话本信息操作

int Hashfunction(int key) //计算哈希函数 采用的是除留取余法

int strcmp(const char \*str1,const char \*str2)

int Trans(char \* a) //数据转换函数 把字符串按ascii码相加的形式转化为数字数据

int Conflict(int i) //冲突处理函数，采用步长为1的线性探测再散列

void GetInfo(char \* inputname, char \* inputtelephone, char \* inputaddress) //从键盘获取联系人信息的操作

void GetName(char \* inputname) //从键盘获取姓名操作

void GetTelephone(char \* inputtelephone) //从键盘获取电话操作

void CreateContact(Sqlist \* telephonehash, Sqlist \* namehash, char newname[], char newtelephone[], char newaddress[])

//创建联系人操作

void DeleteContact(Contact \* now, Sqlist \* namehash, Sqlist \* telephonehash) //删除联系人操作

void SearchName(char \* inputname, Sqlist \* namehash, Sqlist \* telephonehash, Contact \* pnow, char \*name, char \*telephone, char\*address) //按姓名查找联系人操作

void SearchTelephone(char \* inputtelephone, Sqlist \* telephonehash, Sqlist \* namehash, Contact \* pnow, char \*name, char \*telephone, char\*address) //按电话查找联系人操作

void PosiDeleteContact(char \* inputname, Sqlist \* namehash, Sqlist \* telephonehash) //删除联系人操作

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 拥有了以上函数，即可实现电话号码本的功能。如创建联系人，修改联系人，按姓名或按电话查找联系人，此外，还增加了**显示电话号码本信息**和**删除联系人的操作**。另外值得注意的是，在程序的实现中，**并未用到STL标准容器函数**，其中唯一需要用的字符串比较和字符串复制函数可以采用自己编写的形式。如下是部分关键代码：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Hashfunction(int key) //哈希函数运算操作：除留取余法

{

return key%p; //返回哈希函数值（作为数组下标使用）

}

int Conflict(int i) //冲突处理函数 ：使用线性探测再散列的开放地址法

{

i = (i+1)%p;

return i; //返回值为新找的下标值

}

int Trans(char \* a) //把字符串转换为整型并返回该整型数

{

int i = 0;

int s = 0;

while((\*(a+i)) != '\0')

{

s = s + (\*(a+i));

i++;

}

return s;

}

void CreateContact(Sqlist \* telephonehash, Sqlist \* namehash, char newname[], char newtelephone[], char newaddress[]) //创建联系人操作,形参为：新名字 新电话 新地址 Sqlist类型的

{ //两个哈希表

int i = Hashfunction(Trans(newtelephone)); //进行哈希运算，算出下标值（对电话号码进行哈希运算）

int j = Hashfunction(Trans(newname)); //进行哈希运算，算出下标值（对姓名进行哈希运算）

if(telephonehash->nn >= telephonehash->mm) //超出容量，返回值为0，表示出错

{

printf("出错：超出电话本容量！\n"); //提示：超出容量，出错

return; //返回

}

if(telephonehash->nn < telephonehash->mm)

{

while((telephonehash->elem+i)->flag == 1) //判断该位置是否有人，如果有，使用冲突处理函数，直到该位置没有人

{

i = Conflict(i);

}

strcpy( (telephonehash->elem+i)->name, newname); //给名字赋值

strcpy( (telephonehash->elem+i)->telephone, newtelephone); //给电话号码赋值

strcpy( (telephonehash->elem+i)->address, newaddress); //给地址赋值

(telephonehash->elem+i)->flag = 1; //给标志变量赋值，表示有人了

telephonehash->nn++; //现有元素数量+1

}

if(namehash->nn >= namehash->mm) //超出容量，返回值为0，表示出错

{

printf("出错：超出电话本容量！\n"); //提示：超出容量，出错

return; //返回

}

if(namehash->nn < namehash->mm)

{

while((namehash->elem+j)->flag == 1) //判断该位置是否有人，如果有，使用冲突处理函数，直到该位置没有人

{

j = Conflict(j);

}

strcpy( (namehash->elem+j)->name, newname); //给名字赋值

strcpy( (namehash->elem+j)->telephone, newtelephone); //给电话号码赋值

strcpy( (namehash->elem+j)->address, newaddress); //给地址赋值

(namehash->elem+j)->flag = 1; //给标志变量赋值，表示有人了

namehash->nn++; //现有元素数量+1

printf("\n");

printf("创建成功!\n"); //提示：联系人创建成功

printf("\n");

return; //返回

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 下面是程序实现的一些功能的截图：

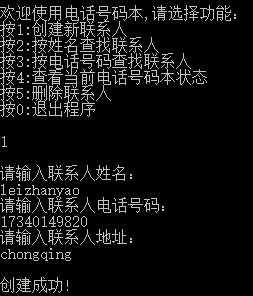


图1.创建联系人操作

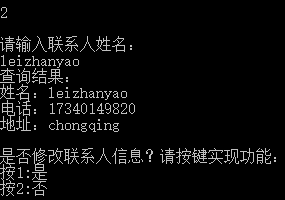


图2.按姓名查找联系人操作

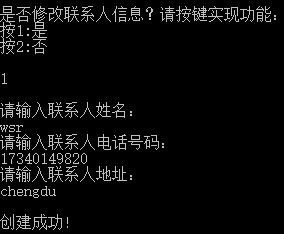


图3.修改联系人操作

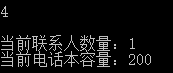


图4.显示当前电话本状态操作

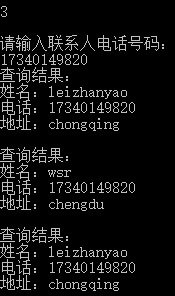


图5.按电话号码查找操作并支持电话重复

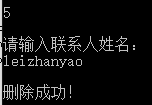


图6.删除联系人操作

**八、总结及心得体会：**

通过本实验的练习，学会了面向对象程序设计的基本方法。

从理论知识层面，更加深了对哈希查找算法的理解，深刻体会了哈希查找在时间复杂度上的优越性，并且体会到了把理论运用于实际中的体验。

在实验操作方面，加深了对于C语言结构体和指针相关知识的理解，掌握了malloc等函数的基本操作。

理论和实际永远都是密不可分的。理论提供了我们解决问题的方法和途径，而实践正是把利用我们研究的理论来解决问题的过程，两者相互依存，密不可分。

**九、对本实验过程及方法、手段的改进建议：**

可以适当增加电话本的其他小功能，比如查看电话本信息，删除联系人，储存信息到文件以便下一次使用等小功能。

**报告评分：**

**指导教师签字：**