**\_\_\_\_2019\_\_\_\_\_\_级\_\_\_\_\_信1901-1\_\_\_\_\_班\_\_\_\_\_2020\_\_年\_\_\_\_11\_\_月\_\_27\_日**

**姓名\_\_\_\_\_李朴凡\_\_\_\_\_\_\_ 学号\_\_\_\_\_\_20194311\_\_\_\_\_\_ 电话\_\_\_\_\_18333135009\_\_\_\_\_\_\_\_**

**1.实验题目**

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。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 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。

**2．需求分析**

需将姓名拼音转化为数字才能比较

哈希表冲突后的探测使用线性探测法

采用数组形式进行存储

**3．概要设计**

存储数据包括 拼音 地址 查找次数，查找长度

哈希表的存储过程

根据关键字查找

函数的调用

**4．详细设计**

typedef struct

{

char \*py; //名字的拼音

int k; //拼音字母得个数

}NAME;

NAME NameList[m]; //全局变量NAME

typedef struct //哈希表

{

char \*py; //名字的拼音

int k; //拼音字母个数

int len; //查找长度

}HASH;

HASH HashList[m]; //全局变量HASH

void CreateHashList() //建立哈希表

{

int i;

for (i = 0; i < m; i++)//初始化

{

HashList[i].py = "";

HashList[i].k = 0;

HashList[i].len = 0;

}

for (i = 0; i < m; i++)

{

int sum = 0;

int key = (NameList[i].k) % M; //哈希函数

int d = key;

if (HashList[key].len == 0) //不冲突

{

HashList[key].k = NameList[i].k;

HashList[key].py = NameList[i].py;

HashList[key].len = 1;

}

else //冲突

{

do

{

d = (d + NameList[i].k % 10 + 1) % M; //伪随机探测再散列法处理冲突

sum = sum + 1; //查找次数加1

} while (HashList[d].k != 0);

HashList[d].k = NameList[i].k;

HashList[d].py = NameList[i].py;

HashList[d].len = sum + 1;

}

}

}

void FindList() //查找

{

int s0 = 0, r, sum = 1, key, d;

printf("\n请输入姓名拼音的关键字:");

scanf\_s("%d", &s0);

key = s0 % M; //使用哈希函数

d = key;

if (HashList[key].k == s0) //分3种情况

printf("\n姓名:%s 关键字:%d 查找长度为: 1", HashList[d].py, s0);

else if (HashList[key].k == 0)

printf("无此记录!");

else

{

int g = 0;

do

{

d = (d + s0 % 10 + 1) % M; //伪随机探测再散列法处理冲突

sum = sum + 1;

if (HashList[d].k == 0)

{

printf("无此记录! ");

g = 1;

}

if (HashList[d].k == s0)

{

printf("\n姓名:%s 关键字:%d 查找长度为:%d", HashList[d].py, s0, sum);

g = 1;

}

} while (g == 0);

}

}

void Display() // 显示哈希表

{

int i;

float average = 0;

printf("\n\n地址\t关键字\t\t搜索长度\tH(key)\t 姓名\n"); //显示的格式

for (i = 0; i < 50; i++)

{

printf("%d ", i);

printf("\t%d ", HashList[i].k);

printf("\t\t%d ", HashList[i].len);

printf("\t\t%d ", HashList[i].k%M);

printf("\t %s ", HashList[i].py);

printf("\n");

}

for (i = 0; i < m; i++)

average += HashList[i].len;

average /= n;

printf("\n\n平均查找长度：ASL(%d)=%f \n\n", n, average);

}

**5．调试分析**

在创建时未加入可以从屏幕输入的创建方式，采用的是在程序创建时已将规定好的数据创建。

**6．使用说明**

1.显示

2.查找

3.退出

选择规定的选项进行相应的操作，对于查找部分是对关键字进行的查找。

**7．测试结果**



