//许家兴 实现二叉树的前序、中序和后序遍历 2018年2月

#include<string>

#include<stdio.h>

#define NAMENUMUMBER 30

#define HASH\_LENGTH 50

#define M 50

typedef struct

{

char \*py;//名字的拼音

int k;//拼音所对应的整数

}NAME;

NAME NameList[30];

typedef struct//哈希表

{

char \*py;//名字的拼音

int k;//拼音所对应的整数

int si;//查找长度

}HASH;

HASH HASHList[50];

void initNameList()//初始化一个有姓名的序列

{

char \*f;

int r, s0, i;

NameList[0].py = "zhanghuimin";

NameList[1].py = "lizhuanhong";

NameList[2].py = "yanghanyu";

NameList[3].py = "guochenchen";

NameList[4].py = "mawenqiu";

NameList[5].py = "zhouzhihui";

NameList[6].py = "xujinyan";

NameList[7].py = "limengke";

NameList[8].py = "luruoshi";

NameList[9].py = "xiehuan";

NameList[10].py = "zhangcan";

NameList[11].py = "kouxiaomeng";

NameList[12].py = "naying";

NameList[13].py = "xulixian";

NameList[14].py = "douyang";

NameList[15].py = "huangjiayang";

NameList[16].py = "shanzhongqiu";

NameList[17].py = "caolibao";

NameList[18].py = "lixudong";

NameList[19].py = "wusen";

NameList[20].py = "mencong";

NameList[21].py = "gaoguangrun";

NameList[22].py = "konghuawei";

NameList[23].py = "liuxiaoyan";

NameList[24].py = "xujiaxing";

NameList[25].py = "wangzichong";

NameList[26].py = "xiaohong";

NameList[27].py = "xiaolan";

NameList[28].py = "xiaoming";

NameList[29].py = "xiaozhuozi";

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

{

s0 = 0;

f = NameList[i].py;

for (r = 0; \*(f + r) != '\0'; r++)

s0 += toascii(\*(f + r));

NameList[i].k = s0;

}

}//初始化一个有姓名的序列

void CreatHashList()

{

int i;

for (i = 0; i<HASH\_LENGTH; i++)

{

HASHList[i].py = " ";

HASHList[i].k = 0;

HASHList[i].si = 0;

}

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

{

int sum = 0;

int adr = (NameList[i].k) % M;

int d = adr;

if (HASHList[adr].si == 0)

{

HASHList[adr].k = NameList[i].k;

HASHList[adr].py = NameList[i].py;

HASHList[adr].si = 1;

}

else

{

do

{

//d = (d + di) % M;

d = (d + NameList[i].k % 10 + 1) % M;

sum = sum + 1;

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

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

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

HASHList[d].si = sum + 1;

}

}

}//用初留余数法构造哈希函数，用线性探测法处理冲突

void FindList()

{

char name[20] = { 0 };

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

printf("请输入姓名的拼音");

scanf\_s("%s", name);

for (r = 0; r<20; r++)

s0 += name[r];

adr = s0%M;

d = adr;

if (HASHList[adr].k == s0)

printf("姓名%s 关键字%d 查找长度1", HASHList[d].py, s0);

else if (HASHList[adr].k == 0)

{

printf("无此记录");

}

else

{

int g = 0;

do

{

d = (d + s0 % 10 + 1) % M;

sum = sum + 1;

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

{

printf("无此记录");

g = 1;

}

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

{

printf("姓名%s 关键字%d 查找长度%d", HASHList[d].py, s0, sum);

g = 1;

}

} while (g == 0);

}

}//构造查询函数，用户输入姓名，系统根据函数进行寻找

void display()

{

int i;

float average = 0;

printf("地址\t 搜索长度\t 姓名\t");

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

{

printf("%d", i);

printf("\t%d", HASHList[i].si);

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

printf("\n");

}

}//输出名字序列

void main()

{

initNameList();

CreatHashList();

display();

FindList();

}//主函数