/\* 实验项目名称:电话号码查询系统的实现

实验目的与要求:

1.基础知识:掌握数据结构中的查找、排序等算法相关知识;

掌握C或VC++语言中程序设计的方法。

2.参考教材相关算法，完成以下程序功能:

(1)自选存储结构实现电话号码表的初始化；

(2)编写一个电话号码查询系统,要求有电话号码记录的录入(插入)存储、查询、记录删除、排序、打印等模块;

实验性质: 综合性(4学时)

说明: 存储结构可采用哈希表的方式,完成用电话号码或姓名为关键字构建哈希表,并进行查询、添加、删除、打印记录等功能模

块(此方式为推荐方式),其次子函数的调用顺序由最终用户决定(可用多分支结构),程序中应有用户的操作选择界面. \*/

# include "stdio.h"

# include "stdlib.h"

# define SUCCESS 1

# define NULL\_KEY -2 //-2为无标志记录

# define UNSUCCESS 0

# define DUPLICATE -1

int hashsize[]={11,19,29,37}; //哈希表容量递增表,一个合适的素数序列

int m=0; //哈希表表长,全局变量

typedef int KeyType; //设关键字为整形

typedef struct {

char name; //姓名

KeyType num; //号码

}Node;

typedef struct {

Node \*elem; //数据元素存储地址,动态分配数组

int count; //当前数据元素个数

int sizeindex; //hashsize[H.sizeindex]为当前容量

}HashTable;

void ChuangJian(HashTable &H) { //构建一个空哈希表

int i;

H.count=0; //当前元素个数

H.sizeindex=0; //初试存储容量为hashsize[0]

m=hashsize[0];

H.elem=(Node \*)malloc(m\*sizeof(Node));

if (!H.elem)

exit (-2); //存储分配失败

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

H.elem[i].num=NULL\_KEY; //未填记录的标志

}

void DaYin(int p,Node e) { //打印下标为p的记录

printf ("哈希下标=%d 姓名:%c 号码:%d\n",p,e.name,e.num);

}

int HaXi(KeyType K) { //哈希函数(m为表长,全局变量)

return K%m;

}

void ChongTu(int &p,int d) { //线性探测再散列

p=(p+d)%m;

}

int EQ(int a,int v) { //判断a和v是否相等

if(a==v)

return 1; //相等返回1

else

return 0; //否则返回0

}

int Find(HashTable H,KeyType K,int &p) {//在开放定址哈希表H中查找关键字为K的元素,若查找成

//功,以p指示待查数据元素在表中下标,并返回SUCCESS;否则,返回UNSUCCESS

int c=0;

p=HaXi(K); //求得哈希地址

while(H.elem[p].num!=NULL\_KEY&&!EQ(K,H.elem[p].num)) {//该位置中填有记录,并且与关键字不相等

c++;

if(c<m)

ChongTu(p,c); //求得下一探查地址p

else

return UNSUCCESS; //查找不成功

}

if (EQ(K,H.elem[p].num))

return SUCCESS; //查找成功,p返回待查数据元素下标

else

return UNSUCCESS; //查找不成功

}

int ChaXun(HashTable H,KeyType K,int &p,int &c) { //在开放定址哈希表中查找关键字为K的元素,若查找成功,以p指示

//待查数据元素在表中位置,并返回SUCCESS;否则,以p指示插入位置,并返回UNSUCCESS;c用以计冲突次数,其初值为0

p=HaXi(K); //求得哈希地址

while(H.elem[p].num!=NULL\_KEY&&!EQ(K,H.elem[p].num)) {//该位置中填有记录,并且与关键字不相等

if(c<m)

ChongTu(p,c); //求得下一探查地址p

else

break;

}

if (EQ(K,H.elem[p].num))

return SUCCESS; //查找成功,p返回待查数据元素下标

else

return UNSUCCESS; //若查找不成功,p返回插入位置

}

void ChongJian(HashTable &H) { //重建哈希表

int ChaRu(HashTable &H,Node e);

int i,count=H.count;

Node \*p,\*elem=(Node \*)malloc(count\*sizeof(Node));

p=elem;

printf ("重建哈希表!\n");

for(i=0;i<m;i++) //保存原有数据到elem中

if((H.elem+i)->num!=NULL\_KEY)//该单元有数据

\*p++=\*(H.elem+i);

H.count=0;

H.sizeindex++; //增大存储容量

m=hashsize[H.sizeindex];

p=(Node \*)realloc(H.elem,m\*sizeof(Node));

if(!p)

exit(-2); //存储分配失败

H.elem=p;

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

H.elem[i].num=NULL\_KEY; //未填记录的标志化

for(p=elem;p<elem+count;p++)//将原有的数据按照新的表长插入到重建的哈希表中

ChaRu(H,\*p);

}

int ChaRu(HashTable &H,Node e) {//查找不成功时插入数据e到开放定址哈希表H中,并返回1;若冲突次数过大,则重建哈希表

int c=0,p;

if(H.count==m-1) {

H.sizeindex++; //增大存储容量

m=hashsize[H.sizeindex];

H.elem=(Node \*)realloc(H.elem,m\*sizeof(Node));//追加空间

if (!H.elem)

exit (-2); //追加空间失败

}

if (ChaXun(H,e.num,p,c)) { //表中已有与e有相同关键字的元素

printf ("表中已有相同关键字的元素!\n");

return DUPLICATE;

}

else {

if (c<hashsize[H.sizeindex]/2) { //冲突次数c未达到上限

H.elem[p]=e; //插入e

++H.count;

return 1;

}

else {

ChongJian(H); //重建哈希表

return UNSUCCESS;

}

}

}

int BaoCun(HashTable H) {//将哈希表中所有数据保存到phone.txt

FILE \*fp;

int i=0,p;

if((fp=fopen("phone.txt","w+"))==NULL)

return 0;

for (p=0;p<m;p++) { //表中当前记录的个数

if(H.elem[p].num!=NULL\_KEY)

i++;

}

fprintf(fp,"%d\n",i);

for (p=0;p<m;p++) { //将哈希表中所有数据写到phone.txt

if(H.elem[p].num!=NULL\_KEY)

fprintf(fp,"%d %c %d\n",p,H.elem[p].name,H.elem[p].num);

}

fclose(fp);

printf ("保存成功!\n");

return 1;

}

int DuQu(HashTable &H) {//将phone.txt所有数据读到哈希表中

FILE \*fp;

int i,p,n;

char a[15];

if((fp=fopen("phone.txt","r+"))==NULL)

return 0;

fscanf(fp,"%s",a);

n=atoi(a); //phone.txt中当前记录的个数

H.count=n;

for (i=0;i<n;i++) {//将phone.txt所有数据读到哈希表中

fscanf(fp,"%s",a);

p=atoi(a);

fscanf(fp,"%s",a);

H.elem[p].name=a[0];

fscanf(fp,"%s",a);

H.elem[p].num=atoi(a);

}

fclose(fp);

return 1;

}

void QingKong(HashTable &H) {//清空哈希表

int i;

H.count=0; //当前元素个数置为0

H.sizeindex=0; //初试存储容量为hashsize[0]

m=hashsize[0];

H.elem=(Node \*)malloc(m\*sizeof(Node));

if (!H.elem)

exit (-2); //存储分配失败

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

H.elem[i].num=NULL\_KEY; //未填记录的标志

printf ("哈希表已清空!\n");

}

void PaiXu(HashTable &H1,HashTable H) {//排序

int i,j;

Node e;

H1.count=H.count;

H1.sizeindex=H.sizeindex;

m=hashsize[0];

H1.elem=(Node \*)malloc(m\*sizeof(Node));

if (!H1.elem)

exit (-2);

for (i=0;i<m;i++) //将哈希表H中所有数据赋给哈希表H1

H1.elem[i]=H.elem[i];

for (i=0;i<m-1;i++) //在哈希表H1中按姓名排序

for (j=i+1;j<m;j++)

if(H1.elem[i].num!=-2&&H1.elem[j].num!=-2)

if(H1.elem[i].name>H1.elem[j].name) {

e=H1.elem[i];

H1.elem[i]=H1.elem[j];

H1.elem[j]=e;

}

}

int main () { //主函数

int i,j,p;

char c,v;

Node e;

KeyType K;

HashTable H,H1;

ChuangJian(H);

DuQu(H);

while(1) {

system("cls");

system("color F0");

printf ("\t|===========================================================|\t\n");

printf ("\t| |\t\n");

printf ("\t| 电话号码系统 |\t\n");

printf ("\t| |\t\n");

printf ("\t|===========================================================|\t\n");

printf ("\n\t【1】-->打印!\t\t【2】-->查询!\t\t【3】-->插入!\n");

printf ("\n\t【4】-->修改!\t\t【5】-->删除!\t\t【6】-->排序!\n");

printf ("\n\t【7】-->保存!\t\t【8】-->清空!\t\t【0】-->退出!\n");

printf ("\n\n\t\t\t请输入你的选择:");

scanf ("%d",&i);

switch(i) {

case 1: //打印

if(H.count>0) {

printf ("按哈希表下标遍历结果如下:\n");

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

if(H.elem[p].num!=NULL\_KEY)

DaYin(p,H.elem[p]);

}

else

printf ("哈希表为空,不用打印!\n");

system("pause");

break;

case 2: //查询

do{

printf ("请输入你要查询的关键字:");

scanf ("%d",&K);

j=Find(H,K,p);

if(j==SUCCESS) {

printf ("你输入的关键字查询结果如下:\n");

DaYin(p,H.elem[p]);

}

else

printf ("该哈希表中没有你要查找的关键字!\n");

printf ("是否继续查找?是/否,y/n:");

getchar();

}while((c=getchar())=='y'||(c=getchar())=='Y');

system("pause");

break;

case 3: //插入

do{

printf ("请输入你要插入的姓名 号码:");

getchar();

scanf ("%c %d",&e.name,&e.num);

ChaRu(H,e);

printf ("是否继续插入?是/否,y/n:");

getchar();

}while((c=getchar())=='y'||(c=getchar())=='Y');

system("pause");

break;

case 4: //修改

if(H.count>0)

do{

printf ("请输入你要修改的关键字:");

scanf ("%d",&K);

j=Find(H,K,p);

if(j==SUCCESS) {

printf ("你输入的关键字查询结果如下:\n");

DaYin(p,H.elem[p]);

printf ("请输入修改后的姓名 号码:");

getchar();

scanf ("%c %d",&e.name,&e.num);

if(e.num!=K) {

H.elem[p].num=NULL\_KEY;

ChaRu(H,e);

}

else

H.elem[p]=e;

}

else

printf ("该哈希表中没有你要修改的关键字!\n");

printf ("是否继续修改?是/否,y/n:");

getchar();

}while((c=getchar())=='y'||(c=getchar())=='Y');

else

printf ("哈希表为空,无法修改!\n");

system("pause");

break;

case 5: //删除

if(H.count>0)

do{

printf ("请输入你要删除的关键字:");

scanf ("%d",&K);

j=Find(H,K,p);

if(j==SUCCESS) {

printf ("你输入的关键字查询结果如下:\n");

DaYin(p,H.elem[p]);

printf ("是否确认删除?是/否,y/n:");

getchar();

if((v=getchar())=='y'||(v=getchar())=='Y') {

H.elem[p].num=NULL\_KEY;

H.count=H.count-1;

printf ("该记录已删除!\n");

}

}

else

printf ("该哈希表中没有你要删除的关键字!\n");

printf ("是否继续删除?是/否,y/n:");

getchar();

}while((c=getchar())=='y'||(c=getchar())=='Y');

else

printf ("哈希表为空,无法删除!\n");

system("pause");

break;

case 6: //排序

if(H.count>0) {

PaiXu(H1,H);

printf ("排序后的哈希表按下标遍历结果如下:\n");

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

if(H1.elem[p].num!=NULL\_KEY)

DaYin(p,H1.elem[p]);

}

else

printf ("哈希表为空,无法排序!\n");

system("pause");

break;

case 7: //保存

BaoCun(H);

system("pause");

break;

case 8: //清空

if(H.count>0)

QingKong(H);

else

printf ("哈希表为空,不用清空!\n");

system("pause");

break;

case 0: //退出程序

printf ("退出程序!\n");

return 1;

default: //重新输入选择的功能序号

printf ("输入有误,请重新输入!\n");

system("pause");

break;

}

}

}