1. 详细设计

#include<stdio.h>

#include<stdlib.h>

#define Maxsize 100

#define NULLK -1 //定义空关键字值

#define DELETE -2 //定义被删除关键字

typedef char \* InfoType;

typedef int KeyType;

typedef struct

{

KeyType key;

InfoType data;

int count;

} HashData;

typedef HashData HashTable[Maxsize];

void InsertHash(HashTable ha,int &n,KeyType k,int p) //插入关键字k

{

int i,d;

d=k % p;

if (ha[d].key==NULLK || ha[d].key==DELETE)

{

ha[d].key=k;

ha[d].count=1;

}

else

{

i=1;

do

{

d=(d+1) % p;

i++;

}

while (ha[d].key!=NULLK && ha[d].key!=DELETE);

ha[d].key=k;

ha[d].count=i;

}

n++;

}

void CreateHash(HashTable ha,KeyType x[],int n,int m,int p) //创建哈希表

{

int i,n1=0;

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

{

ha[i].key=NULLK;

ha[i].count=0;

}

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

InsertHash(ha,n1,x[i],p);

}

int SearchHash(HashTable ha,int p,KeyType k) //查找关键字k

{

int i=0,d;

d=k % p;

while (ha[d].key!=NULLK && ha[d].key!=k) //线性探查法

{

i++;

d=(d+1) % p;

}

if (ha[d].key==k)

return d;

else

return NULLK;

}

int DeleteHash(HashTable ha,int p,int k,int &n) //删除关键字k

{

int d;

d=SearchHash(ha,p,k);

if (d!=NULLK)

{

ha[d].key=DELETE;

n--;

return 1;

}

else

return 0;

}

void DispHash(HashTable ha,int n,int m) //输出哈希表

{

float avg=0;

int i;

printf(" 哈希表地址:\t");

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

printf(" %3d",i);

printf(" \n");

printf(" 哈希表关键字:\t");

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

if (ha[i].key==NULL || ha[i].key==DELETE)

printf(" ");

else

printf(" %3d",ha[i].key);

printf(" \n");

printf(" 探寻次数:\t");

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

if (ha[i].key==NULL || ha[i].key==DELETE)

printf(" ");

else

printf(" %3d",ha[i].count);

printf(" \n");

}

int main() //主函数

{

int x[]= {16,74,60,43,54,90,46,31,29,88,77};

int n=11,m=13,p=13,i,k=29;

HashTable ha;

CreateHash(ha,x,n,m,p);

printf("\n");

DispHash(ha,n,m);

i=SearchHash(ha,p,k);

if (i!=-1)

printf(" ha[%d].key=%d\n",i,k);

else

printf(" 未找到%d\n",k);

k=77;

printf(" 删除关键字%d\n",k);

DeleteHash(ha,p,k,n);

DispHash(ha,n,m);

return 0;

}

四、调试分析

1. 程序使用说明
2. 本程序的运行环境为 Dev cpp；
3. 测试结果

