**实验五 查找**

一、实验目的

1、掌握查找表、动态查找表、静态查找表和平均查找长度的概念。

2、掌握线性表中顺序查找和折半查找的方法。

3、学会哈希函数的构造方法，处理冲突的机制以及哈希表的查找。

二、实验预习

说明以下概念

1、顺序查找：

2、折半查找：

3、哈希函数：

4、冲突及处理：

三、实验内容和要求

1. 静态查找表技术

依据顺序查找算法和折半查找算法的特点，对下面的两个查找表选择一个合适的算法，设计出完整的C源程序。并完成问题：

查找表1 ： { 8 ，15 ，19 ，26 ，33 ，41 ，47 ，52 ，64 ，90 } ，查找key = 41

查找表2 ： {12 ，76 ，29 ，15 ，62 ，35 ，33 ，89 ，48 ，20 } ，查找key =35

查找key=41的算法： 比较次数：

查找key=35的算法： 比较次数：

顺序查找算法算法实现代码

int sxcz(SSTable s ,int k ,int \*p){

int n = 1;

int i;

for(i = s.length;s.data[i]!=k&&i>0;i--)

n++;

\*p=n;

return i;

}

折半查找算法算法实现代码

int zbcz(SSTable s, int k ,int \*p){

int n = 0;

int low , high , mid;

low = 1;

high = s.length;

while(low<=high){

n++;

mid = (low + high) / 2;

if(s.data[mid]==k){

\*p = n;

return mid;

}

else if(s.data[mid]>k){

high = mid - 1;

}

else

low = mid + 1;

}

\*p = n;

return 0;

}

2、哈希表的构造与查找

/\* 采用开放地址法构造哈希表\*/

#include<stdio.h>

#include<malloc.h>

#define MAXSIZE 25

#define P 13

#define OK 1

#define ERROR 0

#define DUPLICATE -1

#define TRUE 1

#define FALSE 0

typedef struct{ /\*哈希表元素结构\*/

int key; /\*关键字值\*/

int flag; /\*是否存放元素\*/

}ElemType;

typedef struct {

ElemType data[MAXSIZE];

int count; /\*元素个数\*/

int sizeindex; /\*当前哈希表容量\*/

}HashTable;

int d1[15]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14}; /\*线性探测序列\*/

int d2[15]={0,1,-1,2\*2,-2\*2,3\*3,-3\*3,4\*4,-4\*4,5\*5,-5\*5,6\*6,-6\*6,7\*7,-7\*7}; /\*二次探测序列\*/

void dataset(int ds[],int \*len);

int InsertHash(HashTable \*H,int e,int d[]);

int CreateHash(HashTable \*H,int ds[],int len,int d[]);

int SearchHash(HashTable \*H, int e,int d[]);

void menu();

/\*输入查找表\*/

void dataset(int ds[],int \*len){

int n,m;

n=0;

printf("\n查找表输入：");

while(scanf("%d",&m)==1){ /\*以输入一个非整数作为结束\*/

ds[n]=m;

n++;

}

\*len=n;

}

/\*计算哈希地址，插入哈希表\*/

int InsertHash(HashTable \*H,int e,int d[]){

int k,i=1;

k=e%P;

while(H->data[k].flag==TRUE||k<0){

k=(e%P+d[i])%MAXSIZE;i++;

if(i>=15)

return ERROR;

}

H->data[k].key=e;

H->data[k].flag=TRUE;

H->count++;

return OK;

}

/\*构造哈希表\*/

int CreateHash(HashTable \*H,int ds[],int len,int d[]){

int i;

for(i=0;i<len;i++){

if(SearchHash(H,ds[i],d)!=-1)

return DUPLICATE;

InsertHash(H,ds[i],d);

if(H->count>=MAXSIZE)

return ERROR;

}

return OK;

}

/\*初始化哈希表\*/

void InitHash(HashTable \*H){

int i;

for(i=0;i<MAXSIZE;i++){

H->data[i].key=0;

H->data[i].flag=FALSE;

}

}

/\*在哈希表中查找\*/

int SearchHash(HashTable \*H, int e,int d[]){

int k,i=1;

k=e%P;

while(H->data[k].key!=e){

k=(e%P+d[i])%MAXSIZE;i++;

if(i>=15)

return -1;

}

return k;

}

/\*演示菜单\*/

void menu(){

int choice;int \*p;

HashTable h;

h.count=0;h.sizeindex=MAXSIZE;

int a[MAXSIZE]={0};

int i,n,e;

dataset(a,&n); /\*建立查找表\*/

getchar();

printf("\n");

do{

printf("\n----哈希查找演示----\n");

printf("\n1.线性探测构造哈希表\n");

printf("\n2.二分探测构造哈希表\n");

printf("\n3.退出\n");

printf("\n输入选择：");

scanf("%d",&choice);

if(choice==1)

p=d1;

else if(choice==2)

p=d2;

else

return;

InitHash(&h); /\*初始化哈希表\*/

if(!(i=CreateHash(&h,a,n,p))) /\*构造哈希表\*/

printf("\n哈希表构造失败！\n");

else if(i==DUPLICATE)

printf("\n哈希表具有重复关键字！\n");

else{

printf("\n哈希表：\n");

for(i=0;i<h.sizeindex;i++)

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

printf("\n\n哈希查找\n输入要查找的key值：");

getchar();

scanf("%d",&e);

if((i=SearchHash(&h,e,p))==-1)

printf("\n%d未找到\n",e);

else

printf("\n%d在哈希表中下标为%d\n",e,i);

}

getchar();

}while(1);

}

int main(){

menu();

return 0;

}

输入查找表为：19 14 23 1 68 20 84 27 55 11 10 79（注意以输入一个非整数结束）。

运行结果：

1)线性探测散列：

哈希表形态：

84在哈希表中的位置:

2）二次探测散列：

哈希表形态：

84在哈希表中的位置：

四、实验小结