完成日期：2016.12.1

* **问题描述：** 试编写利用折半查找确定记录所在块的分块查找算法。

提示：读入各记录建立主表；

按L个记录/块建立索引表；（L=6）

查找；

测试实例：查找K=48， K=13，K=62

* **算法思想：**

索引表中有两部分：索引项和块的数目。索引项中有两部分：该块的最大值和该块的起始位置。

先把数据存到一个线性表中，再分块，并求出每块最大值及起始位置，存在索引项中。从索引表中折半查找索引项，确定块后再从该块中遍历查找。

* **设计描述：**

1）typedef struct {

int maxkey;//子表最大值

int stadr;//第一个元素所在位置

}indexItem,\*inPtr;// 索引项

typedef struct {

indexItem elem[20];//每块起始位置和最大元素 (0不用)

int blocknum;//块的数目

}indexTable,\*TablePtr;//索引表

2）//用来初始化索引表，main调用

int Initial(int Seq[MAX\_SEQ],TablePtr L,int Bcount,int TotalNum)

//知道块后，找块中的元素，Binary函数调用

void Find(int Seq[MAX\_SEQ],TablePtr L,int AD,int SNum,int B1count)

//折半查找算法，main调用

void Binary(int Seq[MAX\_SEQ],TablePtr L,int head,int tail,int Snum,int Bcount)

3）int Initial(int Seq[MAX\_SEQ],TablePtr L,int Bcount,int TotalNum)//初始化索引表

{ //求块的数目

If(TotalNum%Bcount==0)

块的数目=TotalNum/Bcount;

Else

块的数目=TotalNum/Bcount+1；//不够也要加一个块

初始化块0，设为{0，0}

初始化块1，设为{Seq[0]，1}

For(p从1到数据总数)

{ if 是新的块

起始地址置为p+1;

最大置为Seq[p];

if 此块已满

开启新的块;

Else

If（Seq[p]>该块最大值）

最大值=Seq[p];

}

}

void Find(int Seq[MAX\_SEQ],TablePtr L,int AD,int SNum,int B1count)//在块中查找

{ Num=该块起始位置；

遍历该块元素

If（Snum==Seq[count]）

Flag置为1

打印位置；

If(flag==0)

找不到；

}

void Binary(int Seq[MAX\_SEQ],TablePtr L,int head,int tail,int Snum,int Bcount)//二分查找

{mid=(head+tail)/2;

If(mid==0)

找不到

If(head<tail)

{if(查找数字<中间)

Tail=mid；

Binary函数；

If（查找数字>中间）

Head=mid+1；

Binary函数；

Else

寻找该块}

Else 寻找head块

}

* **源程序**

（索引表.c）

#include<stdio.h>

#include<stdlib.h>

#define MAX\_SEQ 100

typedef struct {

int maxkey;//子表最大值

int stadr;//第一个元素所在位置

}indexItem,\*inPtr;// 索引项

typedef struct {

indexItem elem[20];//每块起始位置和最大元素 (0不用)

int blocknum;//块的数目

}indexTable,\*TablePtr;//索引表

int Initial(int Seq[MAX\_SEQ],TablePtr L,int Bcount,int TotalNum)//初始化索引表

{ int p,q;

int i=1;

int BlockNum;//块的数目 （后面计数用）

if(TotalNum%Bcount==0)//求块的数目

{ L->blocknum=TotalNum/Bcount;

}

else

{ L->blocknum=TotalNum/Bcount+1;

}

BlockNum=1;

//标号为0的块不用，设为0,0

//标号为1的块初始化

L->elem[1].stadr=1;

L->elem[1].maxkey=Seq[0];

L->elem[0].stadr=0;

L->elem[0].maxkey=0;

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

{ if(i==1)

{

L->elem[BlockNum].stadr=p+1; //存储初始地址

L->elem[BlockNum].maxkey=Seq[p];//先把该块第一个数设为最大数

}

if(i==Bcount)

{

if(L->elem[BlockNum].maxkey<Seq[p])

{ L->elem[BlockNum].maxkey=Seq[p];

}//开始没加，会遗漏最后一个数据

i=1;

BlockNum++;//开始处理下一块

}

else

{ if(L->elem[BlockNum].maxkey<Seq[p])

{ L->elem[BlockNum].maxkey=Seq[p];

}//找块的最大值

i++;

}

}

printf("\n");

/\*printf("%d ",L->blocknum);

printf("%d",BlockNum);

for(i=1;i<=L->blocknum;i++)

{ printf("\ni->Maxkey=%d i->Stadr=%d \n",L->elem[i].maxkey,L->elem[i].stadr);

}\*/

}

void Find(int Seq[MAX\_SEQ],TablePtr L,int AD,int SNum,int B1count)//知道块后，找块中的元素

{ int count;

int Num,flag=0;

Num=L->elem[AD].stadr;//起始位置

for(count=Num-1;count<Num-1+B1count;count++)

//遍历该块元素 （开始NUM忘了-1，所以不会查找第一个元素，因为Num从1开始计，但是Seq从0开始存）

{ if(SNum==Seq[count])

{ printf("找到了！在第%d块中，位置是%d\n",AD,count+1);

flag=1;

}

}

if(flag==0)//没有找到

{ printf("找不到！\n");

}

}

void Binary(int Seq[MAX\_SEQ],TablePtr L,int head,int tail,int Snum,int Bcount)//折半查找

{ int mid;

mid=(head+tail)/2;//取中间位置

if(mid==0)

{ printf("找不到\n");

}

if(head<tail)

{ if(Snum<L->elem[mid].maxkey)

Binary(Seq,L,head,mid,Snum,Bcount);

else if(Snum>L->elem[mid].maxkey)

Binary(Seq,L,mid+1,tail,Snum,Bcount);

else

Find(Seq,L,mid,Snum,Bcount);

}

else

Find(Seq,L,head,Snum,Bcount);

//底下的算法不算二分查找

/\*if(Snum>L->elem[mid-1].maxkey&&Snum<=L->elem[mid].maxkey)

//当所找元素小于等于中间块的Max，且大于 前一块的Max，表示范围在此块中

{

Find(Seq,L,mid,Snum,Bcount);//调用函数寻找

}

if(Snum>L->elem[mid].maxkey)//当所找元素大于中间块的Max，表示在这之后的块中

{ Binary(Seq,L,mid+1,tail,Snum,Bcount);

}

if(Snum<=L->elem[mid-1].maxkey)//小于等于前一块的Max，表示在这之前的块中

{ Binary(Seq,L,head,mid,Snum,Bcount);

}\*/

}

main()

{ int numCount,block;

int i;

TablePtr LA;

LA=(TablePtr)malloc(sizeof(indexTable));

int seq[MAX\_SEQ]={0};

printf("请输入存储的数字个数：");

scanf("%d",&numCount);

printf("请输入每块数字个数：");

scanf("%d",&block);

printf("请输入要存储的顺序表：\n");

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

{ scanf("%d",&seq[i]);

}

Initial(seq,LA,block,numCount);

int Findnum;

while(1)//可以重复查找数字

{

printf("请输入要查找的数字：");

scanf("%d",&Findnum);

Binary(seq,LA,1,LA->blocknum,Findnum,block);

}

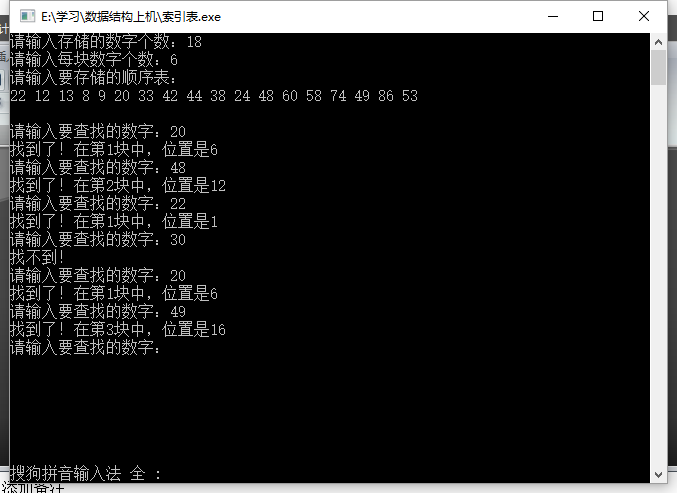
system("pause");

return 0;

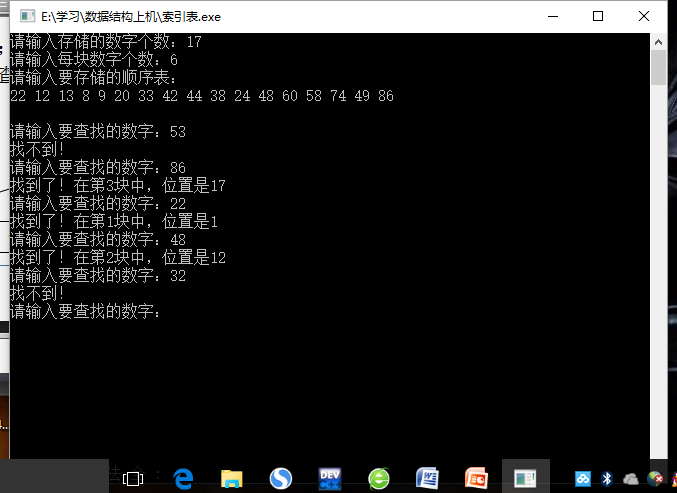
}

* **测试结果：**

1.恰好每块都满了



2.最后一块未满时



* **用户使用说明：**

运行环境Dev C++，注意输入时每块的最大值应按递增顺序排列。块内无序，块间按递增顺序排列。存储的顺序表数据个数应和之前输入的相符。且每块内个数不能大于数据总个数。

* **心得体会：**

开始时我写的折半查找不是实际意义上的折半查找，因为考虑到存的是最大值，所以担心会漏掉查找的块，虽然也能运行成功，但不太符合要求。所以从网上查了下别人写的折半查找的算法，通过head>tail的判断，否则查找head的块，可以避免我担心的问题。

因为存的时候是从Seq[0]开始存的，块是从1开始存的，后来就写乱了，所以查找时总是查不到第一个元素。之后写时应该统一一下，要不都从0开始，要不都从1开始，避免写乱。