算法设计与分析实验一

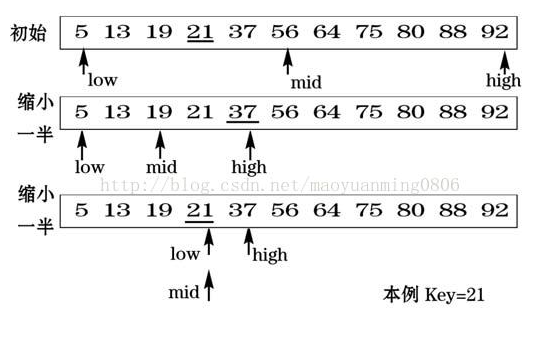
一．实验要求：分别用循环和递归调用方式实现二分查找，语言不限。

二．实验知识补充：

二分查找算法思想：有序的序列，每次都是以序列的中间位置的数来与待查找的关键字进行比较，每次缩小一半的查找范围，直到匹配成功。

例子：将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功；否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表。重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

二分查找图示说明：



二分查找优缺点：

优点是比较次数少，查找速度快，平均性能好；

其缺点是要求待查表为有序表，且插入删除困难。

因此，折半查找方法适用于不经常变动而查找频繁的有序列表。

使用条件：查找序列是顺序结构，有序。

实验代码：

1.递归实现：

#include "stdio.h"

int BinarySearch(int target,int head,int rear);

int num[11]={1,2,3,4,5,6,7,8,9,10,11};

int main()

{

int target,ans;

scanf("%d",&target);

ans = BinarySearch(target,0,11);

if(ans == 0)printf("NO");

else if(ans != 0)printf("YES,下标为%d",ans);

return 0;

}

int BinarySearch(int target,int head,int rear)

{

int middle;middle=(head+rear)/2;

if(rear==head)

{

if(num[middle] == target) return 1;

else return 0;

}

if(num[middle]>target)

{

BinarySearch(target,head,middle-1); //这里需要注意，这里必须传middle-1，不能是middle，

//num[middle]已经大于target了，所以再传过去没有意义，

//直接传middle-1就行了。如果穿middle会产生bug，最后一个数字会

//检测不到，因为除2是取整运算，如果middle不-1会死循环，可以自己

//手写一下就明白了

}

else if(num[middle]<target)

{

BinarySearch(target,middle+1,rear); //middle+1同上

}

else if(num[middle]==target)

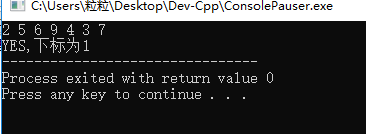
{

return middle;

}

}

实验结果：



2.循环实现:

#include "stdio.h"

int BinarySearch(int num[],int len,int target);

int num[11]={1,2,3,4,5,6,7,8,9,10,11};

int main()

{

int target,ans;

scanf("%d",&target);

ans = BinarySearch(num,11,target);

if(ans == 0)printf("NO");

else if(ans != 0)printf("YES,下标为%d",ans);

return 0;

}

int BinarySearch(int num[],int len,int target)

{

int head=0,rear=len,middle;

while(head<=rear)

{

middle=(head+rear)/2;

if(num[middle]<target) head=middle+1;

else if(num[middle]>target) rear=middle-1;

else return middle;

}

return 0;

}

 实验结果：

