中北大学软件学院

**实验报告**

|  |  |
| --- | --- |
| 专 业： | 软件工程 |
| 课程名称： | 数据结构与算法 |
| 班 级： | 21130420 |
| 学 号： | 2113042015 |
| 姓 名： | 程楚晋 |
| 辅导教师： | 马巧梅 |

2022年03月制

成绩：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验时间 | 2022年06月 1 日7 时至9 时 | 学时数 | 2学时 | |
| 1.实验名称  查找 | | | | |
| 2.实验目的  （1）掌握静态查找表算法（重点掌握折半查找）；  （2）掌握动态查找表——二叉排序树查找算法； | | | | |
| 3.实验内容  **基本内容：**  算法1：采用顺序存储结构创建静态查找表，对查找表进行顺序查找和改进的顺序查找，并对其查找效率进行比较；  算法2：采用顺序存储结构创建静态查找表——有序表，对有序表进行二分查找；  **选作内容：**  编程实现按二叉排序树算法进行查找。 | | | | |
| 4.实验原理  （1）有序表的折半查找过程是：先确定待查记录所在的范围（区间），然后逐步缩小范围直到找到或找不到该记录为止；  （2）二叉排序树查找过程是：首先将给定值和根结点的关键字比较，若相等，则查找成功，否则将依据给定值和根结点的关键字之间的大小关系，分别在左子树或右子树上继续进行查找； | | | | |
| 5.实验源代码  //  // Created by 86178 on 2022/6/1.  //  #include<iostream>  #include "cstdio"  #include<algorithm>  #include<cstring>  #include<malloc.h>  #include<cstdlib>  using namespace std;  typedef int KeyType;  typedef float InfoType;  #define OK 1  #define ERROR 0  #define OVERFLOW -1  #define MAXSIZE 100  #define EQ(a,b) ((a)==(b))  #define LT(a,b) ((a) < (b))  #define LQ(a,b) ((a)<=(b))  using namespace std;  const int N = 11;  typedef struct{  KeyType key;  InfoType otherinfo;  }ElemType;  typedef struct{  ElemType \*elem;  int length;  int listsize;  }SSTable;  int InitList\_Sq(SSTable &L)  { //构造一个空的线性表  L.elem=(ElemType \*)malloc(MAXSIZE\*sizeof(ElemType));  if (! L.elem) exit(0); //存储分配失败  L.length=0; //空表长度为0  L.listsize = N; //初始存储容量  return OK;  }  //线性表的顺序表插入操作  int ListInsert\_Sq(SSTable &L,int i ,ElemType e){  int j;  if(i<1 || i>L.length+1) return ERROR; //i值不合法  if(L.length==L.listsize) return ERROR;  for(j=L.length-1;j>=i-1;j--)  L.elem[j+1]=L.elem[j]; //插入位置及之后的元素后移  L.elem[i-1] = e; //将新元素e放入第i个位置  ++L.length; //表长增1  return OK;  }  //普通顺序查找  int Search\_Seq(SSTable ST,float key)  {  for(int i=ST.listsize;i>=1;--i)  {  //cout<<i<<endl;  if(ST.elem[i].key == key) return i;  }  return 0;  }  //设置监视哨的顺序查找  int Search\_sep(SSTable ST,float key)  {  ST.elem[0].key = key;  for(int i = ST.length;ST.elem[i].key != key; --i )  if(ST.elem[i].key == key) return i;  }  //二分查找  int Search\_Bin(SSTable ST,KeyType key)  {  int low = 1;  int high = ST.length;  int mid;  while(low <= high)  {  mid = low + high >> 1;  if(key == ST.elem[mid].key) return mid;  else if(key < ST.elem[mid].key) high = mid -1;  else low = mid + 1;  }  return 0;  }  int main()  {  SSTable L;  int i;  int y,z,h;  ElemType x;  //创建顺序表  InitList\_Sq(L);  for(i=1;i<=10;i++){  scanf("%d",&x);  ListInsert\_Sq(L,i,x);  }  cout<<"创建的顺序表为"<<endl;  for(i=1;i<=L.length;i++)  printf("%d ",L.elem[i-1].key);  puts("");  cout<<"普通顺序查找(输入需要查找的数)"<<endl;  scanf("%d",&y);  int a = Search\_Seq(L,y);  cout<<a<<endl<<"设置监视哨的顺序查找(输入需要查找的数)"<<endl;  scanf("%d",&z);  int b = Search\_sep(L,z);  cout<<b<<endl;  puts("二分查找");  scanf("%d",&h);  int c = Search\_Bin(L,h);  cout<<c;  return 0;  }  运行结果： | | | |
| 6.实验结论及心得  通过这次实验，让我理解了查找的原理。同时，让我明白了前面的知识对后序学习的重要性，在本次实验中：我遇见了以下的问题:顺序表的创建，因为和第二章学习的顺序表不一样，导致凯斯创建的时候，各种数据类型总是出错，但是经过琢磨，发现，本质其实差不多，因此，学习数据结构，要活学活用，不能太过于死板，程序没有唯一标准，只要理解透彻了各种数据结构和算法的本质，就可以学习起来游刃有余。 | | | |