**《数据结构综合设计》实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 人工智能与大数据学院 | 专业 | 虚拟现实技术 | 班级 | 21级3班 | 学生姓名 | 解芸芸 |
| 实验  周次 | 10-11 | 实验  日期 | 2023.5.4 | 学时 | 4 | 教师姓名 | 李昊康 |
| 项目名称 | | 查找的应用 | | | | | |
| 实验  类别 | 🗹验证型实验 🞎设计型实验 🞎综合型实验 🞎其它 | | | | | 成绩：95 | |
| 1. 实验目的及具体要求   实验目的：  1. 熟练掌握二叉排序树的构造和查找方法。  2. 熟练掌握静态查找表及哈希表查找方法。  具体要求：   1. 从空的二叉树开始，每输入一个结点数据，就建立一个新结点插入到当前已生成的二叉排序树中。 2. 在二叉排序树中查找某一结点。   二、仪器设备和工具  装有并能运行VS2019的电脑。  三、实验内容、步骤及实验数据记录  **1.二叉排序树的构造和查找代码**  #include <stdio.h>  #include <stdlib.h>  #define TRUE 1  #define FALSE 0  #define ENDKEY 0  typedef int KeyType;  typedef struct node  {  KeyType key ; /\*关键字的值\*/  struct node \*lchild,\*rchild;/\*左右指针\*/  }BSTNode, \*BSTree;  int InsertBST(BSTree \*bst, KeyType key)  /\*若在二叉排序树中不存在关键字等于key的元素，插入该元素\*/  {  BSTree s;  if(\*bst==NULL)  {  s=(BSTree)malloc(sizeof(BSTNode));  s->key=key;  s->lchild=NULL;  s->rchild=NULL;  \*bst=s;  }  else if(key<(\*bst)->key)  InsertBST(&((\*bst)->lchild),key);  else if(key>(\*bst)->key)  InsertBST(&((\*bst)->rchild),key);  return 0;  //请完成本函数的功能  }  void CreateBST(BSTree \*bst)  /\*从键盘输入元素的值，创建相应的二叉排序树\*/  {  KeyType key;  \*bst=NULL;  scanf("%d",&key);  while(key!=ENDKEY)  {  InsertBST(bst,key);  scanf("%d",&key);  }  //请完成本函数的功能  }  void InOrder(BSTree bst)  /\*中序遍历二叉树, root为指向二叉树(或某一子树)根结点的指针\*/  {  if (bst!=NULL)  {  InOrder(bst ->lchild); /\*中序遍历左子树\*/  printf("%d->",bst->key); /\*访问根结点\*/  InOrder(bst ->rchild); /\*中序遍历右子树\*/  }  }  BSTree SearchBST(BSTree bst, KeyType key)  /\*在根指针bst所指二叉排序树中，递归查找某关键字等于key的元素，若查找成功，返回指向该元素结点指针，否则返回空指针\*/  {  if(!bst)  return NULL;  else if(bst->key==key)  return bst;  else if(bst->key>key)  return SearchBST(bst->lchild,key);  else  return SearchBST(bst->rchild,key);  //请完成本函数的功能  }  int DelBST(BSTree t, KeyType k) /\*在二叉排序树t中删去关键字为k的结点\*/  {  BSTNode \*p, \*f,\*s ,\*q;  p=t;  f=NULL;  while(p) /\*查找关键字为k的待删结点p\*/  {  if(p->key==k ) break; /\*找到则跳出循环\*/  f=p; /\*f指向p结点的双亲结点\*/  if(p->key>k)  p=p->lchild;  else  p=p->rchild;  }  if(p==NULL) return 0; /\*若找不到，返回原来的二叉排序树\*/  if(p->lchild==NULL) /\*p无左子树\*/  {  if(f==NULL)  t=p->rchild; /\*p是原二叉排序树的根\*/  else  if(f->lchild==p) /\*p是f的左孩子\*/  f->lchild=p->rchild ; /\*将p的右子树链到f的左链上\*/  else /\*p是f的右孩子\*/  f->rchild=p->rchild ; /\*将p的右子树链到f的右链上\*/  free(p); /\*释放被删除的结点p\*/  }  else /\*p有左子树\*/  {  q=p;  s=p->lchild;  while(s->rchild) /\*在p的左子树中查找最右下结点\*/  {  q=s;  s=s->rchild;  }  if(q==p)  q->lchild=s->lchild ; /\*将s的左子树链到q上\*/  else  q->rchild=s->lchild;  p->key=s->key; /\*将s的值赋给p\*/  free(s);  }  return 1;  } /\*DelBST\*/  void main()  {  BSTree T,p;  int keyword,temp;  char ch,j='y';  T=NULL;  while(j!='n')  {  printf("1.创建二叉排序树\n");  printf("2.显示排序的数据\n");  printf("3.查找数据\n");  printf("4.在查找树中插入一个数据\n");  printf("5.在查找树中删除一个数据\n");  printf("6.程序结束，退出\n");  scanf(" %c",&ch); //输入操作选项  switch(ch)  {  case '1':  printf("请输入数据，以0作为数据输入结束。\n");  CreateBST(&T);  break;  case '2':  if(!T) printf("二叉排序树中没有数据。\n");  else {InOrder(T);printf("\b\b \n");}  break;  case '3':  printf("输入待查找的数据值:\n");  scanf("%d",&keyword); //输入要查找元素的关键字  p=SearchBST(T, keyword);  if(!p) printf("%d 没有找到。\n",keyword); //没有找到  else printf("%d 查找成功。\n",keyword); //成功找到  break;  case '4':  printf("输入待插入的数据:");  scanf("%d",&keyword); //输入要插入元素的关键字  temp=InsertBST(&T, keyword);  if(temp==FALSE)  printf("%d 已在二叉树中!\n",keyword); //该元素已经存在  else  printf("%d 插入成功！\n",keyword);  break;  case '5':  printf("输入待删除的数据:");  scanf("%d",&keyword); //输入要删除元素的关键字  temp=DelBST(T, keyword);  if(temp==FALSE) printf("%d 不存在!\n",keyword); //该元素不存在  else printf("成功删除%d\n",keyword); //成功删除  break;  default:  j='n';  }  }  printf("程序结束!\nPress any key to shut off the window!\n");  getchar();  getchar();  }  **2.静态查找表代码**  #include <conio.h>  #include <stdio.h>[文件包含]  #define MAX 30 //定义有序查找表的最大长度  typedef struct{  char elem[MAX]; //有序查找表  int length; //length指示当前有序查找表的长度  }SSTable;  void initial(SSTable &); //初始化有序查找表  int search(SSTable,int); //在有序查找表中查找元素  void print(SSTable); //显示有序查找表中所有元素  void main()  {SSTable ST; //ST为一有序查找表  int ch,loc,flag=1;  char j;  initial(ST); //初始化有序查找表[初始化一个有序查找表ST]  while(flag)  { printf("请选择：\n");  printf("1.显示所有元素\n");  printf("2.查找一个元素\n");  printf("3.退出\n");  scanf(" %c",&j);  switch(j)  {case '1':print(ST); break; //显示所有元素  case '2':{printf("请输入要查找的元素：");  scanf("%d",&ch); //输入要查找的元素的关键字  loc=search(ST,ch); //查找  if(loc!=0) printf("该元素所在位置是：%d\n",loc); //显示该元素位置  else printf("%d 不存在!\n",ch);//当前元素不存在  break;  }  default:flag=0;  }  }  printf("程序运行结束!按任意键退出!\n");  }  void initial(SSTable &v)  {//初始化有序查找表  int i;  printf("请输入静态表的元素个数："); //输入有序查找表初始化时的长度  scanf("%d",&v.length);  printf("请从小到大输入%d个元素（整形数）：\n",v.length);  getchar();  for(i=1;i<=v.length;i++) scanf("%d",&v.elem[i]);  }  int search(SSTable v,int ch)  {//在有序查找表中查找ch的位置，成功返回其位置，失败返回0  int low,high,mid;  low=1;high=v.length; //置区间初值  while(low<=high)  {mid=(low+high)/2;  if(v.elem[mid]==ch) return mid; //找到待查元素  else if(v.elem[mid]>ch) high=mid-1; //继续在前半区间进行查找  else low=mid+1; //继续在后半区间进行查找    }  return 0; //找不到时，i为0  }  void print(SSTable v) //显示当前有序查找表所有元素  {int i;  for(i=1;i<=v.length;i++) printf("%d ",v.elem[i]);  printf("\n");  }  **3.哈希表查找代码**  #include "stdio.h"  #include "stdlib.h"    #define HASHSIZE 10 // 定义散列表长度  #define NULLKEY -32768    typedef struct  {  int \*elem; // 数据元素存储地址，动态分配数组  int count; // 当前数据元素个数  }HashTable;    int m = 0;    int Init(HashTable \*H)  {  int i;    m = HASHSIZE;  H->elem = (int \*)malloc(m \* sizeof(int)); //分配内存  H->count = m;  for (i = 0; i<m; i++)  {  H->elem[i] = NULLKEY;  }  return 1;  }      int Hash(int k)  {  return k % m;//除留余数法  }      void Insert(HashTable \*H, int k)  {  int addr = Hash(k);  while (H->elem[addr] != NULLKEY)  {  addr = (addr+1) % m;//开放定址法  }  H->elem[addr] = k;  }    int Search(HashTable \*H, int k)  {  int addr = Hash(k); //求哈希地址      while (H->elem[addr] != k)//开放定址法解决冲突  {  addr = (addr+1) % m;    if (H->elem[addr] == NULLKEY || addr == Hash(k))  return -1;  }  return addr;  }    void Result(HashTable \*H)//散列表元素显示  {  int i;  for (i = 0; i<H->count; i++)  {  printf("%d ", H->elem[i]);  }  printf("\n");  }    void main()  {  int i, j, addr;  HashTable H;  int arr[HASHSIZE] = { NULL };    Init(&H);    printf("输入关键字集合:");  for (i = 0; i<HASHSIZE; i++)  {  scanf\_s("%d", &arr[i]);  Insert(&H, arr[i]);  }  Result(&H);    printf("输入需要查找的元素：");  scanf\_s("%d", &j);  addr = Search(&H, j);  if (addr == -1)  printf("元素不存在\n");  else  printf("%d元素在表中的位置是:%d\n", j,addr);    }   1. 实验结果及分析 2. **二叉排序树的构造和查找运行结果**   **1**  **图片1**  **3**  **5**   1. **静态查找表运行结果**   **6**   1. **哈希表查找运行结果**     **4.分析**  本次实验难度不大，对于初次接触树的概念的我来说是一次很好的锻炼机会，让我熟悉了建立二叉树的方法，以及加深了对哈夫曼树的理解，同时，增强了我对于二叉树孩子与双亲的关系的理解，有助于更好的了解二叉树的结构。并熟练掌握静态查找及哈希查找方法。 | | | | | | | |

说明：1. 实验周次：填写实际上课周，如第5-8周上课填“5-8”或第10周上课填“10”。

1. 实验报告各部分内容需详实填写，按实验指导书上的评分标准给出分数。
2. 实验类型参考实验类型说明文件。