**《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 | 1班 | 学生姓名 | 翟家杏 |
| 实验  周次 | 9-10 | 实验  日期 | 2023.5.8 | 学时 | 4 | 教师姓名 | 李昊康 |
| 项目名称 | | 树与二叉树的实现及应用 | | | | | |
| 实验  类别 | 🗹验证型实验 🞎设计型实验 🞎综合型实验 🞎其它 | | | | | 成绩：92 | |
| 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：  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。  在二叉排序树中查找某一结点。   1. 实验仪器、设备和材料   装有并能运行VS2019的电脑。   1. 实验内容、步骤及实验数据记录   二叉排序树  #include <stdio.h>  #include <stdlib.h>  #include <conio.h>  #include <math.h>  enum BOOL  {  False, True  };  typedef struct BiTNode  {  char data;  struct BiTNode\* lchild, \* rchild;  }BiTNode, \* BiTree;  BOOL SearchBST(BiTree, char, BiTree, BiTree&);  BOOL InsertBST(BiTree&, char);  BOOL DeleteBST(BiTree&, char);  void Delete(BiTree&);  void InorderBST(BiTree);  void preorderBST(BiTree);  void postorderBST(BiTree);  void main()  {  BiTree T, p;  char ch, keyword, j = 'y';  BOOL temp;  T = NULL;  while (j != 'n')  {  printf("1.元素插入\n");  printf("2.元素查找\n");  printf("3.中序遍历\n");  printf("4.先序遍历\n");  printf("5.后序遍历\n");  printf("6.元素删除\n");  printf("7.退出\n");  scanf(" %c", &ch);  switch (ch)  {  case '1':printf("请输入要插入的结点关键字(char):");  scanf(" %c", &keyword);  temp = InsertBST(T, keyword);  if (!temp) printf("%c 已经存在!\n", keyword);  else printf("成功插入%c!\n", keyword);  break;  case '2':printf("请输入要搜索的结点关键字(char):");  scanf(" %c", &keyword);  temp = SearchBST(T, keyword, NULL, p);  if (!temp) printf("%c 没有找到!\n", keyword);  else printf("成功找到%c!\n", keyword);  break;  case '3':if (!T) printf("二叉树没有根节点!\n");  else { InorderBST(T); printf("\n"); }  break;  case '4':if (!T) printf("二叉树没有根节点!\n");  else { preorderBST(T); printf("\n"); }  break;  case '5':if (!T) printf("二叉树没有根节点!\n");  else { postorderBST(T); printf("\n"); }  break;  case '6':printf("请输入要删除的结点关键字(char):");  scanf(" %c", &keyword);  temp = DeleteBST(T, keyword);  if (!temp) printf("%c 不存在!\n", keyword);  else printf("成功删除%c\n", keyword);  break;  default: j = 'n';  }  }  printf("程序结束!\n请按任意键关闭窗口!\n");  getchar();  }  void InorderBST(BiTree T)  {  if (T->lchild) InorderBST(T->lchild);  printf("%2c", T->data);  if (T->rchild) InorderBST(T->rchild);  }  void preorderBST(BiTree T)  {  printf("%2c", T->data);  if (T->lchild) InorderBST(T->lchild);  if (T->rchild) InorderBST(T->rchild);  }  void postorderBST(BiTree T)  {  if (T->lchild) InorderBST(T->lchild);  if (T->rchild) InorderBST(T->rchild);  printf("%2c", T->data);  }  BOOL SearchBST(BiTree T, char key, BiTree f, BiTree& p)  {  BOOL tmp1, tmp2;  tmp1 = tmp2 = False;  if (!T) { p = f; return False; }  else if (key == T->data) { p = T; return True; }  else if (key < T->data) tmp1 = SearchBST(T->lchild, key, T, p);  else tmp2 = SearchBST(T->rchild, key, T, p);  if (tmp1 || tmp2) return True;  else return False;  }  BOOL InsertBST(BiTree& T, char e)  {  BiTree p, s;  if (!SearchBST(T, e, NULL, p))  {  s = (BiTree)malloc(sizeof(BiTNode));  s->data = e;  s->lchild = s->rchild = NULL;  if (!p) T = s;  else if (e < p->data) p->lchild = s;  else p->rchild = s;  return True;  }  else return False;  }  BOOL DeleteBST(BiTree& T, char key)  {  BOOL tmp1, tmp2;  tmp1 = tmp2 = False;  if (!T) return False;  else  {  if (key == T->data) { Delete(T); return True; }    else if (key < T->data) tmp1 = DeleteBST(T->lchild, key);  else tmp2 = DeleteBST(T->rchild, key);  if (tmp1 || tmp2) return True;  else return False;  }  }  void Delete(BiTree& p)  {  BiTree s, q;  if (!p->rchild)  {  q = p;  p = p->lchild;  free(q);  }  else if (!p->lchild)  {  q = p;  p = p->rchild;  free(q);  }  else  {  q = p;  s = p->lchild;  while (s->rchild)  {  q = s; s = s->rchild;  }  p->data = s->data;  if (q != p) q->rchild = s->rchild;  else q->lchild = s->lchild;  free(s);  }  }  查找每一结点  #include <conio.h>  #include <stdio.h>[文件包含]  #define MAX 30  typedef struct {  char elem[MAX];  int length;  }SSTable;  void initial(SSTable&);  int search(SSTable, int);  void print(SSTable);  void main()  {  SSTable ST;  int ch, loc, flag = 1;  char j;  initial(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;  case '2': {printf("请输入要查找的元素：");  scanf("%d", &ch);  loc = search(ST, ch);  if (loc != 0) printf("该元素所在位置是：%d\n", loc);  else printf("%d 不存在!\n", ch);  break;  }  default:flag = 0;  }  }  printf("程序运行结束!按任意键退出!\n");  }  void initial(SSTable& v)  {  int i;  printf("请输入静态表的元素个数：");  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)  {  int low, high, mid;  low = 1; high = v.length;  while (low <= high)  {  mid = (low + high) / 2;  if (v.elem[mid] == ch) return mid;  else if (v.elem[mid] > ch) high = mid - 1;  else low = mid + 1;  }  return 0;  }  void print(SSTable v)  {  int i;  for (i = 1; i <= v.length; i++) printf("%d ", v.elem[i]);  printf("\n");  }   1. 实验结果及分析   e40dc511f9e9a47cdd0caac53057fd2  f80d21ea488b437bbbf6de9f3fcac25  e5e54868071f0694899288ccbc91951  1781876c80717356526e31a648af50b  849e4dd9c9ab3b163089f88131ed5f5  分析：  熟练掌握二叉排序树的构造和查找方法  熟练掌握静态查找表及哈希表查找方法 | | | | | | | |

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。