**计算机科学与工程学院实验报告4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验课程名称** | | **实验四 查找和排序应用类实验**  **——二叉排序树应用** | | | **实验成绩** |  |
| **专业** | **计算机科学与技术** | | **班级** | **2206** | **指导教师签字** |  |
| **姓名** | **李昕鸿** | | **姓名** | **黄璐媛** | **姓名** |  |
| **学号** | **20225868** | | **学号** | **20225956** | **学号** |  |
| **附录实验程序及结果：**  **实验分工以及占比：**  **李昕鸿 50%：**利用二叉树实现创建域名二叉树以及删除域名结点  **黄璐媛 50%：**利用二叉树实现插入和查找域名结点  **实验程序：**      **结果：**    **源代码：**  #include<iostream>  #include<stdio.h>  #include<string.h>  #include<stdlib.h>  using namespace **std**;  *//结点 二叉树*  typedef struct **trnode**{       char data[30];       struct **trnode** \*lchild,\*rchild;  }**trnode**,\***tree**;  *//查找树中待插入结点*  int **searchbst**(**tree** T,char \*key,**tree** f,**tree** \*p){       if(!T){            \*p=f;            return 0;       }  *//查找成功*       else if(**strcmp**(key,T->data)==0){            \*p=T;            return 1;       }       else if(**strcmp**(key,T->data)<0){            return **searchbst**(T->lchild,key,T,p);       }       else return **searchbst**(T->rchild,key,T,p);  }  *//查找结点*  int **searchnod**(**trnode** \*N,char \*key){  **trnode** \*M;       M=N;       while(M!=**NULL**&&**strcmp**(M->data,key)!=0){            if(**strcmp**(M->data,key)<0) M=M->rchild;            else M=M->lchild;       }       if(M) cout**<<**"查找失败"**<<endl**;       else cout**<<**"查找成功"**<<endl**;       return 1;  }  *//插入结点*  int **insertbst**(**tree** \*T,char \*key){  **tree** p,s;  *//未查找到*       if(!**searchbst**(\*T,key,**NULL**,&p)){            s=(**trnode**\*)**malloc**(sizeof(**trnode**));  **strcpy**(s->data,key);            s->lchild=s->rchild=**NULL**;  *//插入s*            if(!p){                 \*T=s;            }            else if(**strcmp**(key,p->data)<0){                 p->lchild=s;            }            else p->rchild=s;            return 1;       }       else return 0;  }  *//删除结点*  int **deletenod**(**tree** \*p){  **tree** q,s;       if((\*p)->rchild==**NULL**){            q=\*p;            \*p=(\*p)->rchild;  **free**(q);       }       else if((\*p)->lchild==**NULL**){            q=\*p;            \*p=(\*p)->rchild;  **free**(q);       }  *//左右子树均不空*       else {            q=\*p;            s=(\*p)->lchild;            while(s->rchild){                 q=s;                 s=s->rchild;            }  **strcpy**((\*p)->data,s->data);            if(q!=\*p) q->rchild=s->lchild;            else q->lchild=s->lchild;  **free**(s);       }       return 1;  }  *//查找删除节点*  int **deletebst**(**tree** \*T,char \*key){       if(!\*T) return 0;       else{            if(**strcmp**(key,(\*T)->data)==0) return **deletenod**(T);            else if(**strcmp**(key,(\*T)->data)<0) return **deletebst**(&(\*T)->lchild,key);            else return **deletebst**(&(\*T)->rchild,key);       }  }  *//遍历*  int **ordertr**(**trnode** \*root){       if(!root){ return 0;}  **ordertr**(root->lchild);       cout**<<**root->data**<<endl**;  **ordertr**(root->rchild);       return 1;  }  *//功能*  void **struc**(**trnode** \*A){       int k;       char a[30],c[30],d[30];       cout**<<**"请输入操作\n1-查找\n2-删除\n3-插入\n输入:";       cin**>>**k;       switch(k){  *//查询*            case 1:            {                 cout**<<**"请输入需要查找的节点: ";                 cin**>>**c;  **searchnod**(A,c);                 cout**<<endl**;                 break;            }  *//删除*            case 2:            {                 cout**<<**"请输入需要删除的节点: ";                 cin**>>**a;                 if(!**deletebst**(&A,a)){                      cout**<<**"不存在此节点！"**<<endl**;                 }                 else{                      cout**<<**"删除节点成功"**<<endl**;                      cout**<<**"删除后树的中序遍历结果为: "**<<endl**;  **ordertr**(A);                 }                 break;            }  *//插入*            case 3:            {                 cout**<<**"请输入要插入的节点: ";                 cin**>>**d;                 if(!**insertbst**(&A,d)) cout**<<**"插入失败！要插入的节点已存在"**<<endl**;                 else{                      cout**<<**"插入成功!插入后树的中序遍历结果为: "**<<endl**;  **ordertr**(A);                 }                 break;            }  *//错误输入*            default :            {                 cout**<<**"输入数值错误"**<<endl**;                 break;            }       }  }  int **main**()  {  **trnode** \*A;       A=(**trnode**\*)**malloc**(sizeof(**trnode**));       A->lchild=A->rchild=**NULL**;       int j,k,m=1;       char b[30];       char \*s;  *//创建域名二叉树*       cout**<<**"请输入域名个数: ";       cin**>>**k;       for(j=0;j<k;j++){  *//trnode \*X;*            char \*str;            int i;  *//中转数组*            char zzarray[15];  *//X=A;*            cout**<<**"请输入网址"**<<**j+1**<<**": ";            cin**>>**b;            s=b;  *//反转字符串*  **\_strrev**(s);            while(\*s!='\0'){  *//查找第一个'.'出现的位置 翻转*                 str=**strchr**(s,'.');                 if(str){                      i=str-s;                      zzarray[i+1]='\0';                      while(i>=0){                           zzarray[i]=s[0];                           i--,s++;                      }                 }                 else {  **\_strrev**(s);                      i=**strlen**(s);                      zzarray[i+1]='\0';                      while(i>=0){                           zzarray[i]=s[i];                           i--;                      }                      \*s='\0';                 }  **insertbst**(&A,zzarray);            }       }  *//输出*       cout**<<**"中序遍历结果为: "**<<endl**;  **ordertr**(A);  *//操作*       while(m){  **struc**(A);            cout**<<**"是否继续操作?是-1 不是-0"**<<endl**;            cin**>>**m;       }       cout**<<**"end"**<<endl**;  **system**("pause");       return 0;  } | | | | | | |
| **实验课程总结**  通过本次实验课程内容的学习，我了解了有关排序和查找的相关知识。主要学习的是有关在二叉树中遍历查找，本次实验中利用二叉树以及查找算法实现了二叉排序树应用，基于二叉排序树的搜索互联网域名的程序。本次实验内容提升了我对排序和查找相关知识的了解，加强了有关查找排序的应用。 | | | | | | |

**批改时间：**