实验六 二叉排序树的基本操作

#include <iostream>

using namespace std;

typedef struct BiTNode

{

int data;

struct BiTNode \*lchild,\*rchild;

}BiTNode, \*BiTree;

void Search(BiTree B,int key,BiTree F,BiTree &p)

{

if(!B)

{

p=F;

cout<<"未查找到相应的数"<<endl;

}

else

if(key==B->data)

{

p=B;

cout<<"查找成功"<<endl;

}

else

if(key>B->data)

Search(B->rchild,key,B,p);

else

Search(B->lchild,key,B,p);

}

void Insert(BiTree \*B,int key)

{

BiTree S;

if(\*B==NULL)

{

S=(BiTree)malloc(sizeof(BiTNode));

S->data=key;

S->lchild=NULL;

S->rchild=NULL;

\*B=S;

}

else

if(key > (\*B)->data)

Insert(&((\*B)->rchild),key);

else

if(key < (\*B)->data)

Insert(&((\*B)->lchild),key);

}

void CreateBiTree(BiTree \*B,int \*a,int n)

{

(\*B) = NULL;

for(int i = 0;i < n;i++)

Insert(B,a[i]);

}

void Inorder(BiTree B)

{

if(B==NULL)

return;

Inorder(B->lchild);

cout<<B->data<<" ";

Inorder(B->rchild);

}

void Delet(BiTree &B)

{

BiTree S,Q;

if(B->lchild==NULL)

{

Q=B;

B=B->rchild;

free(Q);

}

else if(B->rchild==NULL)

{

Q=B;

B=B->lchild;

free(Q);

}

else if(B->lchild!=NULL && B->rchild!=NULL)

{

Q=B;

S=B->lchild;

while(S->rchild)

{

Q=S;

S=S->rchild;

}

B->data=S->data;

if(Q!=B)

Q->rchild=S->lchild;

else

Q->lchild=S->lchild;

free(S);

}

}

int Deletprint(BiTree &B,int key)

{

if(!B)

{

cout<<"没有需要删除的数"<<endl;

return 0;

}

else

{

if(key==B->data)

Delet(B);

else

if(key<B->data)

Deletprint(B->lchild,key);

else

Deletprint(B->rchild,key);

}

return 0;

}

void sum(BiTree &B,int &m,int &n,int l)

{

if(B==NULL)

return;

n++;

m=m+l;

sum(B->lchild,m,n,l+1);

sum(B->rchild,m,n,l+1);

}

double length(BiTree &B)

{

int m=0,n=0;

sum(B,m,n,1);

return m\*1.0/n;

}

void main()

{

int n,i,key,x,u;

BiTree B,f,p;

cout<<"请输入节点个数"<<endl;

cin>>n;

cout<<"请输入"<<n<<"个数"<<endl;

int \*a = (int\*)malloc(sizeof(int)\*n);

for(i=0;i<n;i++)

cin>>a[i];

CreateBiTree(&B,a,n);

cout<<"中序遍历序列:"<<endl;

Inorder(B);

cout<<endl;

cout<<"输入需要查找的数:"<<endl;

cin>>key;

Search(B,key,f,p);

cout<<"平均查找长度:"<<endl;

cout<<length(B)<<endl;

cout<<"输入需要插入的数:"<<endl;

cin>>u;

Insert(&B,u);

Inorder(B);

cout<<endl;

cout<<"输入需要删除的数:"<<endl;

cin>>x;

Deletprint(B,x);

cout<<"中序遍历序列:"<<endl;

Inorder(B);

cout<<endl;

}