Binary Search Tree

#include<iostream.h>

#include<conio.h>

#include<process.h>

struct node

{ int data;

node \*left;

node \*right;

node \*parent;

} \*root=NULL, \*temp, \*t, \*a;

int ar[50],ti,prele;

void display(node \*p)

{ if(p!=NULL)

{ cout<<p->data<<" ";

display(p->left);

display(p->right);

}

}

void inorder(node \*p)

{ if(p!=NULL)

{ inorder(p->left);

ar[ti]=p->data;

ti++;

inorder(p->right);

}

}

void preorder(node \*p)

{ if(p!=NULL)

{ if(prele==p->data)

{ a=p;

}

preorder(p->left);

preorder(p->right);

}

}

void insert()

{ temp=new node;

cout<<"\nEnter the data:\t";

cin>>temp->data;

temp->left=NULL;

temp->right=NULL;

temp->parent=NULL;

if(root==NULL)

{ root=temp;

goto fin;

}

t=root;

while(1)

{ if((temp->data)<=(t->data))

{ if((t->left)==NULL)

{ t->left=temp;

temp->parent=t;

goto fin;

}

else

{ t=t->left;

}

}

else if((temp->data)>(t->data))

{ if((t->right)==NULL)

{ t->right=temp;

temp->parent=t;

goto fin;

}

else

{ t=t->right;

}

}

}

fin:

}

void remove()

{ t=root;

int ele,found;

char c='p';

if(root==NULL)

{ cout<<"\nTree doesn't exist!!";

goto fin;

}

cout<<"\nEnter the element you want to delete?\t";

cin>>ele;

found=0;

while(1)

{ if(ele<t->data)

{ if(t->left==NULL)

{ goto end;

}

else

{ c='l';

t=t->left;

}

}

else if(ele>t->data)

{ if(t->right==NULL)

{ goto end;

}

else

{ c='r';

t=t->right;

}

}

else if(ele==t->data)

{ found=1;

goto end;

}

}

end:

if(found==1)

{ if(t->left==NULL)

{ if(t->right==NULL)

{ if(c=='l')

{ a=t->parent;

a->left=NULL;

t->parent=NULL;

delete t;

}

if(c=='r')

{ a=t->parent;

a->right=NULL;

t->parent=NULL;

delete t;

}

if(c=='p')

{ delete t;

root=NULL;

}

goto fin;

}

}

if(t->left==NULL)

{ if(t->right!=NULL)

{ if(c=='l')

{ a=t->parent;

a->left=t->right;

a=t->right;

a->parent=t->parent;

t->parent=NULL;

t->right=NULL;

delete t;

}

if(c=='r')

{ a=t->parent;

a->right=t->right;

a=t->right;

a->parent=t->parent;

t->parent=NULL;

t->right=NULL;

delete t;

}

if(c=='p')

{ root=t->right;

t->right=NULL;

root->parent=NULL;

delete t;

}

goto fin;

}

}

if(t->right==NULL)

{ if(t->left!=NULL)

{ if(c=='l')

{ a=t->parent;

a->left=t->left;

a=t->left;

a->parent=t->parent;

t->parent=NULL;

t->left=NULL;

delete t;

}

if(c=='r')

{ a=t->parent;

a->right=t->left;

a=t->left;

a->parent=t->parent;

t->parent=NULL;

t->left=NULL;

delete t;

}

if(c=='p')

{ root=t->left;

t->left=NULL;

root->parent=NULL;

delete t;

}

goto fin;

}

}

if(t->left!=NULL)

{ if(t->right!=NULL)

{ ti=0;

inorder(root);

int i;

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

{ if(ele==ar[i])

{ i=i-1;

prele=ar[i];

i++;

}

}

preorder(root);

t->data=a->data;

if(a->left!=NULL)

{ temp=a->parent;

temp->right=a->left;

temp=a->left;

temp->parent=a->parent;

a->parent=NULL;

a->left=NULL;

delete a;

}

else

{ temp=a->parent;

temp->left=NULL;

a->parent=NULL;

delete a;

}

goto fin;

}

}

}

if(found==0)

{ cout<<"\nElement not found!!";

}

fin:

}

void main()

{ int ch;

clrscr();

while(1)

{ cout<<"\n\n1)Insert\n2)Delete\n3)Exit\n";

cin>>ch;

if(ch==1)

{ insert();

cout<<"\nBinary Search Tree (Preorder traversal)\n";

display(root);

}

else if(ch==2)

{ remove();

cout<<"\nBinary Search Tree (Preorder traversal)\n";

display(root);

}

else if(ch==3)

{ exit(1);

}

else

{ cout<<"\nWrong choise!!\n\n";

}

}

}

