AVL

#include<iostream.h>

#include<conio.h>

#include<malloc.h>

typedef struct node

{ int data;

struct node \*left,\*right;

int ht;

}node;

node \*insert(node \*,int);

node \*Delete(node \*,int);

void preorder(node \*);

void inorder(node \*);

int height( node \*);

node \*rotateright(node \*);

node \*rotateleft(node \*);

node \*RR(node \*);

node \*LL(node \*);

node \*LR(node \*);

node \*RL(node \*);

int BF(node \*);

void main()

{

clrscr();

node \*root=NULL;

int x,n,i,op;

gotoxy(33,2);

cout<<"AVL"<<endl;

do

{

cout<<"\n1)Insert : ";

cout<<"\n2)Delete : ";

cout<<"\n3)Display: ";

cout<<"\n4)Quit : ";

cout<<"\nEnter Your Choice : ";

cin>>op;

switch(op)

{

case 1:

cout<<"\nEnter a data : ";

cin>>x;

root=insert(root,x);

break;

case 2:

cout<<"\nEnter a data : ";

cin>>x;

root=Delete(root,x);

break;

case 3:

cout<<"\nPreorder sequence :\n";

preorder(root);

cout<<"\nInorder sequence :\n";

inorder(root);

break;

}

}while(op!=4);

}

node \* insert(node \*T,int x)

{

if(T==NULL)

{

T=(node\*)malloc(sizeof(node));

T->data=x;

T->left=NULL;

T->right=NULL;

}

else

if(x > T->data)

{

T->right=insert(T->right,x);

if(BF(T)==-2)

if(x>T->right->data)

T=RR(T);

else

T=RL(T);

}

else

if(x<T->data)

{

T->left=insert(T->left,x);

if(BF(T)==2)

if(x < T->left->data)

T=LL(T);

else

T=LR(T);

}

T->ht=height(T);

return(T);

}

node \* Delete(node \*T,int x)

{

node \*p;

if(T==NULL)

{

return NULL;

}

else

if(x > T->data)

{

T->right=Delete(T->right,x);

if(BF(T)==2)

if(BF(T->left)>=0)

T=LL(T);

else

T=LR(T);

}

else

if(x<T->data)

{

T->left=Delete(T->left,x);

if(BF(T)==-2)

if(BF(T->right)<=0)

T=RR(T);

else

T=RL(T);

}

else

{

if(T->right !=NULL)

{

p=T->right;

while(p->left != NULL)

p=p->left;

T->data=p->data;

T->right=Delete(T->right,p->data);

if(BF(T)==2)

if(BF(T->left)>=0)

T=LL(T);

else

T=LR(T);

}

else

return(T->left);

}

T->ht=height(T);

return(T);

}

int height(node \*T)

{

int lh,rh;

if(T==NULL)

return(0);

if(T->left==NULL)

lh=0;

else

lh=1+T->left->ht;

if(T->right==NULL)

rh=0;

else

rh=1+T->right->ht;

if(lh>rh)

return(lh);

return(rh);

}

node \* rotateright(node \*x)

{

node \*y;

y=x->left;

x->left=y->right;

y->right=x;

x->ht=height(x);

y->ht=height(y);

return(y);

}

node \* rotateleft(node \*x)

{

node \*y;

y=x->right;

x->right=y->left;

y->left=x;

x->ht=height(x);

y->ht=height(y);

return(y);

}

node \* RR(node \*T)

{

T=rotateleft(T);

return(T);

}

node \* LL(node \*T)

{

T=rotateright(T);

return(T);

}

node \* LR(node \*T)

{

T->left=rotateleft(T->left);

T=rotateright(T);

return(T);

}

node \* RL(node \*T)

{

T->right=rotateright(T->right);

T=rotateleft(T);

return(T);

}

int BF(node \*T)

{

int lh,rh;

if(T==NULL)

return(0);

if(T->left==NULL)

lh=0;

else

lh=1+T->left->ht;

if(T->right==NULL)

rh=0;

else

rh=1+T->right->ht;

return(lh-rh);

}

void preorder(node \*T)

{

if(T!=NULL)

{

cout<<T->data<<" Bf="<<BF(T)<<" ";

preorder(T->left);

preorder(T->right);

}

}

void inorder(node \*T)

{

if(T!=NULL)

{

inorder(T->left);

cout<<T->data<<" Bf="<<BF(T)<<" ";

inorder(T->right);

}

}

