TBT

#include <iostream.h>

#include <stdlib.h>

#include <conio.h>

enum marker

{

CHILD,

THREAD

};

struct tbstNode

{

int data;

struct tbstNode \*link[2];

int marker[2];

};

struct tbstNode \*root = NULL;

struct tbstNode \* createNode (int data)

{

struct tbstNode \*newNode;

newNode = (struct tbstNode \*)malloc(sizeof (struct tbstNode));

newNode->data = data;

newNode->link[0] = newNode->link[1] = NULL;

newNode->marker[0] = newNode->marker[1] = THREAD;

return newNode;

}

void insertion(int data)

{

struct tbstNode \*parent, \*newNode, \*temp;

int path;

if (!root)

{

root = createNode(data);

return;

}

parent = root;

while (1)

{

if (data == parent->data)

{

cout<<"Duplicates Not Allowed\n";

return;

}

path = (data > parent->data) ? 1 : 0;

if (parent->marker[path] == THREAD)

break;

else

parent = parent->link[path];

}

newNode = createNode(data);

newNode->link[path] = parent->link[path];

parent->marker[path] = CHILD;

newNode->link[!path] = parent;

parent->link[path] = newNode;

return;

}

void del(int data)

{

struct tbstNode \*current, \*parent, \*temp;

int path;

parent = root;

current = root;

while (1)

{

if (data == current->data)

break;

path = (data > current->data) ? 1 : 0;

if (current->marker[path] == THREAD) {

cout<<"Given data is not available!!\n";

return;

}

parent = current;

current = current->link[path];

}

if (current->marker[1] == THREAD)

{

if (current->marker[0] == CHILD)

{

temp = current->link[0];

while (temp->marker[1] == CHILD)

{

temp = temp->link[1];

}

temp->link[1] = current->link[1];

if (current == root)

{

root = current->link[0];

}

else

{

parent->link[path] = current->link[0];

}

}

else

{

if (current == root)

{

root = NULL;

}

else

{

parent->link[path] = current->link[path];

parent->marker[path] = THREAD;

}

}

}

else

{

temp = current->link[1];

if (temp->marker[0] == THREAD)

{

temp->link[0] = current->link[0];

temp->marker[0] = current->marker[0];

if (temp->marker[0] == CHILD)

{

struct tbstNode \*x = temp->link[0];

while (x->marker[1] == CHILD)

{

x = x->link[1];

}

x->link[1] = temp;

}

if (current == root)

{

root = temp;

}

else

{

cout<<"path:"<<path<<"\ndata:"<<parent->data<<endl;

parent->link[path] = temp;

}

}

else

{

struct tbstNode \*child;

while (1)

{

child = temp->link[0];

if (child->marker[0] == THREAD)

break;

temp = child;

}

if (child->marker[1] == CHILD)

temp->link[0] = child->link[1];

else

{

temp->link[0] = child;

temp->marker[0] = THREAD;

}

child->link[0] = current->link[0];

if (current->marker[0] == CHILD)

{

struct tbstNode \*x = current->link[0];

while(x->marker[1] == CHILD)

x = x->link[1];

x->link[1] = child;

child->marker[0] = CHILD;

}

child->link[1] = current->link[1];

child->marker[1] = CHILD;

if (current == root)

root = child;

else

parent->link[path] = child;

}

}

free(current);

return;

}

void inorder(tbstNode \*temp)

{

if (temp != NULL)

{

inorder(temp->link[0]);

cout<<temp->data;

inorder(temp->link[1]);

}

}

void traversal()

{

struct tbstNode \*myNode;

if (!root)

{

cout<<"Threaded Binary Tree Not Exists!!\n";

return;

}

myNode = root;

while (1)

{

while(myNode->marker[0] == CHILD)

{

myNode = myNode->link[0];

}

cout<<myNode->data;

myNode = myNode->link[1];

if (myNode)

{

cout<<myNode->data;

myNode = myNode->link[1];

}

if (!myNode)

break;

}

cout<<endl;

return;

}

void search(int data)

{

struct tbstNode \*myNode;

int path;

if (!root)

{

cout<<"Tree Not Available!!\n";

return;

}

myNode = root;

while (1)

{

if (myNode->data == data)

{

cout<<"Given data present in TBST!!\n";

return;

}

path = (data > myNode->data) ? 1 : 0;

if (myNode->marker[path] == THREAD)

break;

else

myNode = myNode->link[path];

}

cout<<"Given data is not present in TBST!!\n";

return;

}

int main ()

{

clrscr();

int data, ch;

gotoxy(33,2);

cout<<"TBT"<<endl;

while (1)

{

cout<<"1. Insertion\t2. Deletion\n";

cout<<"3. Searching\t4. Traversal\n";

cout<<"5. Exit\nEnter your choice:";

cin>>ch;

switch (ch)

{

case 1:

cout<<"Enter your input data:";

cin>>data;

insertion(data);

break;

case 2:

cout<<"Enter your input data:";

cin>>data;

del(data);

break;

case 3:

cout<<"Enter your input data:";

cin>>data;

search(data);

break;

case 4:

cout<<"Tree is:";

traversal();

break;

case 5:

exit(0);

default:

cout<<"You have entered wrong option!!\n";

break;

}

cout<<endl;

}

}

