BINARY TREE TRANSVERSAL

ALGORITHM

Step 1: Repeat Steps 2 to 4 while TREE != NULL

Step 2: INORDER(TREE -> LEFT)

Step 3: Write TREE -> DATA

Step 4: INORDER(TREE -> RIGHT)

[END OF LOOP]

Step 5: END

PROGRAM

//traversal in a binary tree

#include<stdio.h>

#include<stdlib.h>

struct node

{

int element;

struct node \*left;

struct node \*right;

};

struct node \*createNode(int val)

{

struct node \*Node=(struct node\*)malloc(sizeof(struct node));

Node->element=val;

Node->left=NULL;

Node->right=NULL;

return(Node);

}

void traversePreorder(struct node \*root)

{

if(root==NULL)

return;

printf("%d\n",root->element);

traversePreorder(root->left);

traversePreorder(root->right);

}

void traversePostorder(struct node \*root)

{

if(root==NULL)

return;

traversePostorder(root->left);

traversePostorder(root->right);

printf("%d\n",root->element);

}

void traverseinorder(struct node \*root)

{

if(root==NULL)

return;

traverseinorder(root->left);

printf("%d\n",root->element);

traverseinorder(root->right);

}

int main()

{

struct node \*root=createNode(36);

root->left=createNode(26);

root->right=createNode(46);

root->left->left=createNode(21);

root->left->right=createNode(31);

root->left->left->left=createNode(11);

root->left->left->right=createNode(24);

root->right->left=createNode(41);

root->right->right=createNode(56);

root->right->right->left=createNode(51);

root->right->right->right=createNode(66);

printf("preoder transversal\n");

traversePreorder(root);

printf("postoder transversal\n");

traversePostorder(root);

printf("inoder transversal\n");

traverseinorder(root);

return 0;

}

//traversal in a binary tree

#include<stdio.h>

#include<stdlib.h>

struct node

{

int element;

struct node \*left;

struct node \*right;

};

struct node \*createNode(int val)

{

struct node \*Node=(struct node\*)malloc(sizeof(struct node));

Node->element=val;

Node->left=NULL;

Node->right=NULL;

return(Node);

}

void traversePreorder(struct node \*root)

{

if(root==NULL)

return;

printf("%d\n",root->element);

traversePreorder(root->left);

traversePreorder(root->right);

}

void traversePostorder(struct node \*root)

{

if(root==NULL)

return;

traversePostorder(root->left);

traversePostorder(root->right);

printf("%d\n",root->element);

}

void traverseinorder(struct node \*root)

{

if(root==NULL)

return;

traverseinorder(root->left);

printf("%d\n",root->element);

traverseinorder(root->right);

}

int main()

{

struct node \*root=createNode(36);

root->left=createNode(26);

root->right=createNode(46);

root->left->left=createNode(21);

root->left->right=createNode(31);

root->left->left->left=createNode(11);

root->left->left->right=createNode(24);

root->right->left=createNode(41);

root->right->right=createNode(56);

root->right->right->left=createNode(51);

root->right->right->right=createNode(66);

printf("preoder transversal\n");

traversePreorder(root);

printf("postoder transversal\n");

traversePostorder(root);

printf("inoder transversal\n");

traverseinorder(root);

return 0;

OUTPUT:

