Iterative Preorder Traversal

Given a Binary Tree, write an iterative function to print Preorder traversal of the given binary tree.

Refer this for recursive preorder traversal of Binary Tree. To convert an inherently recursive procedures to iterative, we need an explicit stack. Following is a simple stack based iterative process to print Preorder traversal.

1) Create an empty stack *nodeStack* and push root node to stack.

2) Do following while *nodeStack* is not empty.

….**a)** Pop an item from stack and print it.

….**b)** Push right child of popped item to stack

….**c)** Push left child of popped item to stack

Right child is pushed before left child to make sure that left subtree is processed first.

C++

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <stack>

using namespace std;

/* A binary tree node has data, left child and right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node with the given data and
   NULL left and right  pointers.*/
struct node* newNode(int data)
{
    struct node* node = new struct node;
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
```

```
}

// An iterative process to print preorder traversal of Binary tree
void iterativePreorder(node *root)
{
    // Base Case
    if (root == NULL)
       return;

    // Create an empty stack and push root to it
    stack<node *> nodeStack;
    nodeStack.push(root);

    /* Pop all items one by one. Do following for every popped item
       a) print it
       b) push its right child
       c) push its left child
    Note that right child is pushed first so that left is processed first */
    while (nodeStack.empty() == false)
    {
        // Pop the top item from stack and print it
        struct node *node = nodeStack.top();
        printf ("%d ", node->data);
        nodeStack.pop();

        // Push right and left children of the popped node to stack
        if (node->right)
            nodeStack.push(node->right);
        if (node->left)
            nodeStack.push(node->left);
    }
}

// Driver program to test above functions
int main()
{
    /* Constructed binary tree is
             10
           /   \
         8      2
       /  \    /
     3     5  2
  */
  struct node *root = newNode(10);
  root->left        = newNode(8);
  root->right       = newNode(2);
  root->left->left  = newNode(3);
  root->left->right = newNode(5);
  root->right->left = newNode(2);
  iterativePreorder(root);
  return 0;
}
```

Run on IDE

Java

```
// Java program to implement iterative preorder traversal
import java.util.Stack;

// A binary tree node
class Node {
```

```
    int data;
    Node left, right;

    Node(int item) {
        data = item;
        left = right = null;
    }
}

class BinaryTree {

    Node root;

    void iterativePreorder()
    {
        iterativePreorder(root);
    }

    // An iterative process to print preorder traversal of Binary tree
    void iterativePreorder(Node node) {

        // Base Case
        if (node == null) {
            return;
        }

        // Create an empty stack and push root to it
        Stack<Node> nodeStack = new Stack<Node>();
        nodeStack.push(root);

        /* Pop all items one by one. Do following for every popped item
         a) print it
         b) push its right child
         c) push its left child
         Note that right child is pushed first so that left is processed first */
        while (nodeStack.empty() == false) {

            // Pop the top item from stack and print it
            Node mynode = nodeStack.peek();
            System.out.print(mynode.data + " ");
            nodeStack.pop();

            // Push right and left children of the popped node to stack
            if (mynode.right != null) {
                nodeStack.push(mynode.right);
            }
            if (mynode.left != null) {
                nodeStack.push(mynode.left);
            }
        }
    }

    // driver program to test above functions
    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(10);
        tree.root.left = new Node(8);
        tree.root.right = new Node(2);
        tree.root.left.left = new Node(3);
        tree.root.left.right = new Node(5);
        tree.root.right.left = new Node(2);
        tree.iterativePreorder();

    }
```

```
}

// This code has been contributed by Mayank Jaiswal
```

Run on IDE

Python

```
# Python program to perform iterative preorder traversal

# A binary tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# An iterative process to print preorder traveral of BT
def iterativePreorder(root):

    # Base CAse
    if root is None:
        return

    # create an empty stack and push root to it
    nodeStack = []
    nodeStack.append(root)

    #  Pop all items one by one. Do following for every popped item
    #   a) print it
    #   b) push its right child
    #   c) push its left child
    # Note that right child is pushed first so that left
    # is processed first */
    while(len(nodeStack) > 0):

        # Pop the top item from stack and print it
        node = nodeStack.pop()
        print node.data,

        # Push right and left children of the popped node
        # to stack
        if node.right is not None:
            nodeStack.append(node.right)
        if node.left is not None:
            nodeStack.append(node.left)

# Driver program to test above function
root = Node(10)
root.left = Node(8)
root.right = Node(2)
root.left.left = Node(3)
root.left.right = Node(5)
root.right.left = Node(2)
iterativePreorder(root)

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Run on IDE

Output:

```
10 8 3 5 2 2
```

This article is compiled by Saurabh Sharma and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

40 Comments Category: Trees

Related Posts:

- Check if removing an edge can divide a Binary Tree in two halves
- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

Recommend 5 Share Sort by Newest

Join the discussion…

Mandy • 20 days ago

No need to store left child

```
Stack<node> nodeStack = new Stack<node>();
nodeStack.push(root);

while (nodeStack.empty() == false) {

   Node mynode = nodeStack.peek();
   nodeStack.pop();

   while(mynode != null){

        System.out.print(mynode.data + " ");

        if  (mynode.right != null) {
              nodeStack.push(mynode.right);
        }
        mynode = mynode.left;

    }
}
```

• Reply • Share ›

surbhijain93 • 2 months ago

```
void iterativePreorder(struct node* root)

{

stack<struct node*=""> s;



while(!s.empty() || root)
```

```
{

s.push(root);

while(root->left)

{
```

see more

• Reply • Share ›

Hiccup • 5 months ago

@Author

Amazing solution. i was trying to solve in some difficult way...I solved how system stack works...

code : https://app.box.com/s/wveurxnx...

• Reply • Share ›

Jex • 5 months ago

http://animatedarena.com/Jex/f...

An animation for tree traversal

• Reply • Share ›

nishant sinha • 7 months ago

stack<node *=""> nodeStack; plz explain this line

• Reply • Share ›

SlickHackz ➚ nishant sinha • 5 months ago

We used Stack container. A container holds object. The object can be anything. In our case, the object the container is going to hold is a tree node. Hence we declare stack<node *=""> S. If we are going to store integers, then the declaration will be stack<int> S

• Reply • Share ›

AkankshaT94 • 9 months ago

Can we do it using queue with the steps as:
temp=root
while(temp)
{1. Print data
2. Enqueue left node
3. Enqueue right node
4. temp=dequeue}

• Reply • Share ›

NIKHIL SINGH ➚ AkankshaT94 • 9 months ago

it will be level order traversing not preorder

2 • Reply • Share ›

Gautham Kumaran • 9 months ago

This can be optimized by pushing only the right nodes to the stack.

```
while(true){
            if(curr!=null){
                System.out.println(curr.data+" ");
                if(curr.right!=null){
                    stack.push(curr.right);
                }
                curr = curr.left;
            }
            else{
                if(!stack.isEmpty()){
                    curr = stack.pop();
                }
                else{
                    break;
                }
            }
        }
```

6 • Reply • Share ›

Holden ➚ Gautham Kumaran • 7 months ago

Thank you for your code. Do you know why my solution which is in Java, is not working? It goes to a infinite loop!

```
        public static void iterativePreorderMine(Node root){
                Stack<node> stack = new Stack<node>();
                Node current = root;
                while(current != null || !stack.isEmpty()){
                        System.out.print(current.data + " ");
                        if(current.right != null){
                                stack.push(current.right);
                        }
                        if(current.left != null){
                                current = current.left;
```

```
                    } else if (!stack.isEmpty()) {
                        current = stack.pop();
                    }
                }
            }
```

• Reply • Share ›

Guest ➚ Holden • 5 months ago

@Holden

Your logic is wrong.

• Reply • Share ›

Narendra ➚ Gautham Kumaran • 8 months ago

agree with you I just coded like the one you wrote and it is similar to iterative in order.

1 • Reply • Share ›

radek • 9 months ago

we can also use the same method specified as www.geeksforgeeks.org/inorder-...
just move the pop statement to after the push statement..
something like..

```
if(head!=NULL) {
push(&st,head);
printf("%d ",head->data);
head=head->left;
}
else {
if(!isempty(st)) {
head=pop(&st);
head=head->right;
}
```

this too works :) :)

2 • Reply • Share ›

Shivani Aggarwal • 10 months ago

This should also work according to me.
http://ideone.com/mLBm2J

• Reply • Share ›

NITIN PANCHAL • a year ago

we can use the following code as well

//to try to implement the iterative preorder traversal in the given tree

```
//to try to implement the iterative preorder traversal in the given tree

#include<iostream>

#include<stdio.h>

#include<stdlib.h>

#include<stack>

#include<queue>

using namespace std ;

struct tree

{

int data ;
```

see more

• Reply • Share ›

Mission Peace • a year ago

https://www.youtube.com/watch?...

My video on this question

1 • Reply • Share ›

Rishabh Mamgain • a year ago

```
void rec(struct node *p)

{

struct node *stack[10];

int top=0;

stack[++top]=0;

while(p!=0)

{ printf("%d",p->info);

if(p->rlink!=0)

{ stack[++top]=p->rlink;

}
```

```
if(p->llink!=())
```

see more

• Reply • Share ›

rishabh Rishabh Mamgain • a year ago

please don't write code in comments

• Reply • Share ›

sree_ec • a year ago

```
void preorderIterative(tree_int* root)
{
mystack* stack=NULL;

if(root == NULL)
return;

while(1)
{
if(root != NULL)
{
printf("%d ",root->data);
push(&stack,root);
root = root->left;
}
else
{
if(stack == NULL)
```

see more

• Reply • Share ›

sk • 2 years ago

```
void Preorder(bst *t)
{
stack<bst *=""> s;
while(1)
{
while(t!=NULL)
{
cout<<t->data<<" ";
s.push(t->rchild);
t=t->lchild;
}
if(s.empty())break;
```

```
else
t=s.pop();
}
}
```

• Reply • Share ›

samthebest • 2 years ago

```
void iterativePreorder(btnode *t)
{
if(t==NULL)
return;

stack <btnode *=""> s;
btnode *current=t;
while(current!=NULL||!s.empty())
{
if(current!=NULL)
{
cout<<current->data<<" ";
s.push(current);
current=current->lchild;
}
else
{ current=s.top();
s.pop();
current=current->rchild;
}
}

}
```

• Reply • Share ›

carmen cojocaru • 2 years ago

Can you post an implementation for the post-order also? I've seen some versions over the Internet but they're pretty complicated. Yours is so clean. Thank you.

1 • Reply • Share ›

Vivek • 2 years ago

no need to push the right child into the stack

```
void preOrderIterative(struct node *root)

{
```

```
        struct stknode *st=NULL;

        while(!empty(st) || root)

        {

                while(root)

                {

                        printf("%d ",root->data);
```

see more

1 • Reply • Share ›

Amit Bgl • 3 years ago

wow code :D

1 • Reply • Share ›

dex • 3 years ago

```
 /*
iterative preorder of bst using explicit stack
*/

#include<stdio.h>
#include<stdlib.h>

struct tree
{
        int data;
        struct tree *left;
        struct tree *right;
};
typedef struct tree node;
void addnode(node *,int);


struct llist
```

see more

2 • Reply • Share ›

dex ➚ dex • 3 years ago

debugged!,

I constructed the tree itself wrong while checking p->data and n .

• Reply • Share ›

dex • 3 years ago

/*
iterative preorder of bst using explicit stack
*/

#include
#include

struct tree
{
int data;
struct tree *left;
struct tree *right;
};
typedef struct tree node;
void addnode(node *,int);

struct llist
{
struct tree *tp;

see more

• Reply • Share ›

zyzz • 3 years ago

i think this one is easy

```
 /void preorder(struct node *temp){
int top=0;
struct node *s[20];
s[0]=NULL;
printf("preorder : \n");
while(temp!=NULL){
    printf("%d \t",temp->data);

    if(temp->right!=NULL){

        s[++top]=temp->right;
    }
```

```
    if(temp->left!=NULL){
```

see more

• Reply • Share ›

abhishek08aug • 3 years ago

Intelligent :D

• Reply • Share ›

SHASHI KUMAR • 3 years ago

```
#include
#include
struct node
{
int data;
struct node *right;
struct node *left;
}*root,*S[10];
int i=0;
void Push(struct node *p)
{
S[i++]=p;
}
struct node *Pop()
{
return S[--i];
}
void preorder()
```

see more

• Reply • Share ›

Veer Verma • 3 years ago

Is modified version of Morris Traversal possible for PreOrder??

• Reply • Share ›

Ashok • 3 years ago

Using stack is as good as using recursion. Modify the morris algorithm from inorder traversal to preorder or post order to get the intended answer.

3 • Reply • Share ›

Ashok • 3 years ago

Using stack is as good as using recursion. Modify the morris algorithm from inorder

traversal to preorder or post order to get the intended answer.

```
/* Paste your code here (You may delete these lines if not writing code) */
```


Venki • 3 years ago

I guess the code can be refactored. Preorder relatively consumes less stack space. In the above code it pushes both the nodes to stack, which is not necessary. See sample code (not tested),

```
void Preorder(Node *pRoot) {
   Node *pMove = pRoot;
   stack<Node *> s;

   s.push(NULL); // To recognize end of processing
   while( pMove ) {

       cout << pMove->key;

       if( pMove->right )
           s.push(pMove->right); // Only we need to keep track of right sub-trees

       if( pMove->left )
           pMove = pMove->left;
       else {
           pMove = s.top();
           s.pop();
       }
   }
}
```


Palash ➚ Venki • 3 years ago

You'd at best be saving one node space in the stack, that you are anyway using by saving that NULL in the beginning. No point of this.

Venki ➚ Palash • 3 years ago

It is nothing but do-undo beheviour. It consumes processing power for no value addition. Pushing one NULL doesn't cost much when compared to repeated do-undo actions.

Suman • 3 years ago

```
[sourcecode language="JAVA"]
public static void traverseIterative(TreeNode root){
Stack<TreeNode> stack = new Stack<TreeNode>();
stack.push(root);
TreeNode currentNode = root;
while(!stack.isEmpty()){
while (currentNode != null){
System.out.println(currentNode.data);
currentNode = currentNode.left;
if (currentNode != null){
stack.push(currentNode);
}
}
currentNode = stack.pop();
currentNode = currentNode.right;
if (currentNode != null){
stack.push(currentNode);
}
}
}
```

• Reply • Share ›

Leet • 3 years ago

Can we do preorder traversal without recursion and without stack?

• Reply • Share ›

Vikas ➚ Leet • 3 years ago

That would be using Morris Traversal. Look it up on wiki.

1 • Reply • Share ›

atul ➚ Vikas • 3 years ago

morris traversal does inorder traversal

```
/* Paste your code here (You may delete these lines if not writing code
```

• Reply • Share ›