**Depth First traversal**

There are three types of depth-first traversal: pre-order, in-order and post-order. For a binary tree, they are defined as operations recursively at each node, starting with the root node as follows:

**Pre-order**

1. Visit the root.
2. Traverse the left subtree.
3. Traverse the right subtree.

**In-order**

1. Traverse the left subtree.
2. Visit the root.
3. Traverse the right subtree.

**Post-order**

1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the root.

**V-visiting node L-traversing left R-traversing right**

VLR Preorder tree traversal

LVR Inorder tree traversal

LRV Postorder tree traversal

//non-recursive

void prerder(node \*leaf){

if (leaf != NULL)

{

push(leaf,stck);

while ( stack not empty){

leaf = pop(stck);

printf(“%d “,leaf->data);

if ( leaf->right != NULL)

push (leaf->right,stck);

if (leaf->left\_child != NULL)

push (leaf->left,stck);

}

}

}

**// A utility function to print preorder traversal of the tree.**

void preOrder(struct node \*root)

{

if(root != NULL)

{

printf("%d ", root->key);

preOrder(root->left);

preOrder(root->right);

}

}

**// A utility function to print inorder traversal of the tree.**

void inOrder(struct node \*root)

{

if(root != NULL)

{

inOrder (root->left);

printf("%d ", root->key);

inOrder (root->right);

}

}

**// A utility function to print postorder traversal of the tree.**

void postOrder(struct node \*root)

{

if(root != NULL)

{

postOrder (root->left);

postOrder (root->right);

printf("%d ", root->key);

}

}

7

3

2 | 6

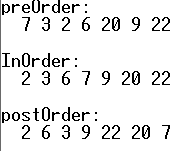
PreOrder Example 7 3 2 6

|  |  |  |  |
| --- | --- | --- | --- |
| po(7)  printf(7);  po(3);  po(null) | po(3)  printf(3);  po(2);  po(6) | po(2)  printf(2);  po(null);  po(null) | po(null)  } |
| po(null)  } |
| po(2)  printf(6);  po(null);  po(null) | po(null)  } |
| po(null)  } |
| po(null)  } | | |

7

3 20

2 | 6 9 | 22

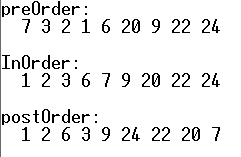


7

3 20

2 | 6 9 | 22

1 | | 24



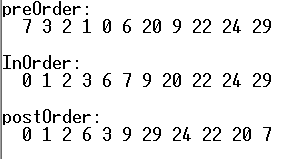
7

3 20

2 | 6 9 | 22

1 | | 24

0 | | 29



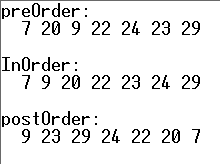
7

20

9 | 22

| 24

23 | 29



7

3 20

2 | 6

1 |

0 |

-5

-9 -3

