BinaryTreeDisplay

*Recursive Display*



1. *LDR - Left\_Root\_Right*

void LDR ( BinaryTreeNode \* root )

{

if ( root == NULL )

return;

LDR ( root -> left );

print ( root -> value );

LDR ( root -> right );

return;

}

1. *DLR - Root\_Left\_Right*

void DLR ( BinaryTreeNode \* root )

{

if ( root == NULL )

return;

print( root -> value );

DLR ( root -> left );

DLR ( root -> right );

return;

}

1. *LRD - Left\_Right\_Root*

void LRD ( BinaryTreeNode \* root )

{

if ( root == NULL )

return;

LRD ( root -> left );

LRD ( root -> right );

print ( root -> value );

return;

}

*Non - Recursive Display - Stack*

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |



1. *LDR - Left\_Root\_Right*

*Sequence of Btree - 4, 2, 5, 1, 6, 3, 7*

1. *DLR - Root\_Left\_Right*
2. *LRD - Left\_Right\_Root*



*Round 1 - Access Root 1 in the Tree, and push the Root Node into stack.*

|  |
| --- |
|  |
|  |
| 1 |

*Output: Empty*

*Round 2 - Check whether Internal Node 2 of Root 1 equals to NULL, push the Internal Node 2 into stack if not NULL.*

|  |
| --- |
|  |
| *2* |
| 1 |

*Output: Empty*

*Round 3 - Check whether Leaf Node 4 of Internal Node 2 equals to NULL, push the Internal Node 4 into stack of not NULL.*

|  |
| --- |
|  |
|  |
| *4* |
| *2* |
| 1 |

*Output: Empty*

*Round 4 - Leaf Node 4 has no Left Node, but the stack has element. Print the current Node Value in the top of stack, and pop out the Leaf Node 4. Assign NULL value to Node.*

|  |
| --- |
|  |
|  |
|  |
| *2* |
| 1 |

*Output: 4*

*Round 5 - Visit Top Node Element 2 of stack, and print the Element Value of Node 2. Pop the Internal Node 2, and pop out the Internal Node 2. The next step is to visit Right Node 3 of Node 2.*

|  |
| --- |
|  |
|  |
|  |
|  |
| 1 |

*Output: 4 -> 2*

*Round 6 - Visit Right Node 3 of the Last Node 2. Push the Right Node 3 into the stack.*

|  |
| --- |
|  |
|  |
|  |
| *3* |
| 1 |

*Output: 4 -> 2*

*Round 7 - Visit the Left Node 6 of Node 3, and push the Node 6 into the stack.*

|  |
| --- |
|  |
|  |
| *6* |
| *3* |
| *1* |

*Output: 4 -> 2*

*Round 8 - Visit the Top Element in the Stack, and print the value of the Top Element 6 out. Pop the element 6 out of the stack. Assign NULL to the Node since Node 6 has not the Right Node.*

|  |
| --- |
|  |
|  |
|  |
| *3* |
| *1* |

*Output: 4 -> 2 -> 6*

*Round 9 - Visit the Next Top Element in the Stack, and print the value 3 out. Pop the Element Node 3 out of the stack. The next step is to visit the Right Node 7 of Node 3.*

|  |
| --- |
|  |
|  |
|  |
|  |
| *1* |

*Output: 4 -> 2 -> 6 -> 3*

*Round 10 - Visit the Right Node 7 of Node 3. Push Node 7 into stack.*

|  |
| --- |
|  |
|  |
|  |
| *7* |
| *1* |

*Output: 4 -> 2 -> 6 -> 3*

*Round 11 - The Right Node 7 has no Left Node and Right Node, so visit the Node 7, print the Node Value 7. Pop the Right Node 7 out of stack.*

|  |
| --- |
|  |
|  |
|  |
|  |
| *1* |

*Output: 4 -> 2 -> 6 -> 3 -> 7*

*Round 12 - The Left Node of Node 7 is NULL. Therefore access the First Top Node 1 in the stack and print the value of Node 1. Pop out the First Top Node 1 and the stack would turn into Empty.*

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

*Output: 4 -> 2 -> 6 -> 3 -> 7 -> 1*

*void LRD ( BinaryTreeNode \* root )*

*{*

*if ( root == NULL )*

*return;*

*stack < BinaryTreeNode \* > s;*

*BinaryTreeNode \* node = root;*

*while ( node | | ! s.empty ( ) )*

*{*

*if ( node )*

*{*

*s.push\_back ( node );*

*node = node -> left;*

*}*

*else*

*{*

*node = s.top( );*

*print ( node -> value );*

*s.pop( );*

*node = node -> right;*

*}*

*}*

*}*

1. *Level*

*void BtreeLevelDisplay ( BinaryTreeNode \* root )*

*{*

*if ( root == NULL )*

*return;*

*stack < BinaryTreeNode \* > s;*

*s.push\_back ( root );*

*while ( s.top( ) )*

*{*

*BinaryTreeNode \* node = s.top( );*

*print( s.top( ) -> value( ) );*

*if ( s.top() -> left )*

*s.push\_back( s.top() ->left );*

*if ( s.top() -> right )*

*s.push\_back( s.top() -> right );*

*s.pop( );*

*}*

*}*