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*

*void LRD ( BinaryTreeNode \* root )*

*{*

*}*

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( );*

*}*

*}*