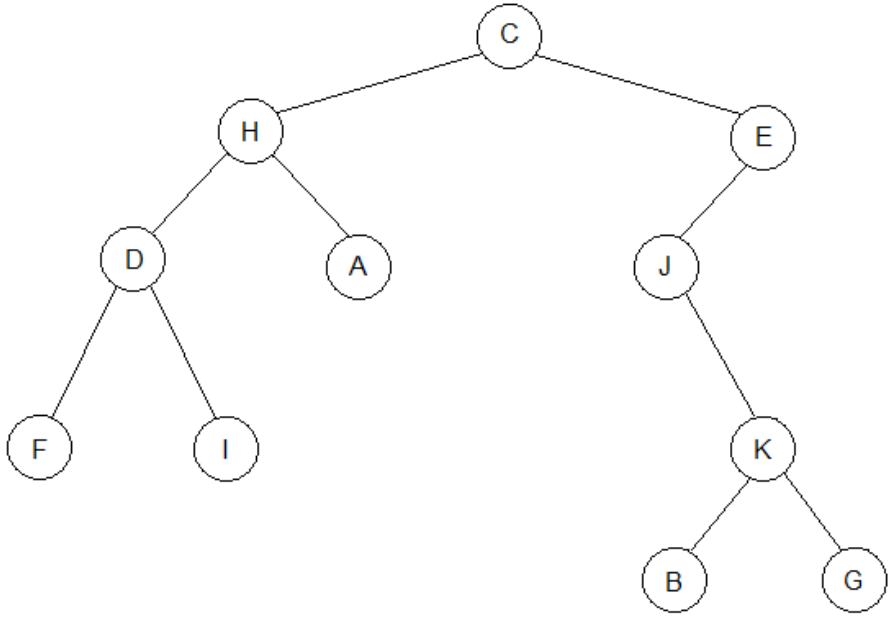
6. Trees :: Binary Trees :: Traversals :: Breadth First

In a breadth first traversal, we visit all of the nodes at level l before moving to level $l + 1$, starting with the root node at level 0. For a tree data structure, this type of traversal is referred to as a **level order traversal**. During a level order traversal of our example tree, we would visit the nodes in this order:

The standard algorithm for performing a level order traversal uses a queue to store nodes to be visited. Remember, that a queue is a first-in-first-out data structure, so nodes must be added to the queue in the order in which we wish to visit them.



6. Trees :: Binary Trees :: Traversals :: Breadth First (continued)

Here is the standard algorithm:

```
-- Performs a level order traversal of the subtree rooted at pRoot. pVisitor is an object which
-- implements a method named visit() which will be called as each Node is visited.
Method levelOrderTraversal(In: Node pRoot, In: pVisitor) Returns Nothing
   Create a queue of Nodes named nodeQueue

   -- Add pRoot to the queue. This will be the first Node that is visited.
   nodeQueue.enqueue(pRoot)

   -- Continue looping while there are still more Nodes to visit.
   While nodeQueue is not empty Do
      -- Get the next Node to be visited from the front of the queue. Visit the Node by calling
      -- the visit() method on pVisitor and passing the data stored in the Node.
      root ← nodeQueue.dequeue()
      pVisitor.visit(root.data)

      -- If root has a left child then that is the Node that will be visited after we visit
      -- root. Add the left child to nodeQueue.
      nodeQueue.enqueue(root.leftChild)

      -- If root has a right child then that is the Node that will be visited after we visit the
      -- left child of root (assuming the left child exists). Add the right child to nodeQueue.
      nodeQueue.enqueue(root.rightChild)
   End While
End Method levelOrderTraversal
```