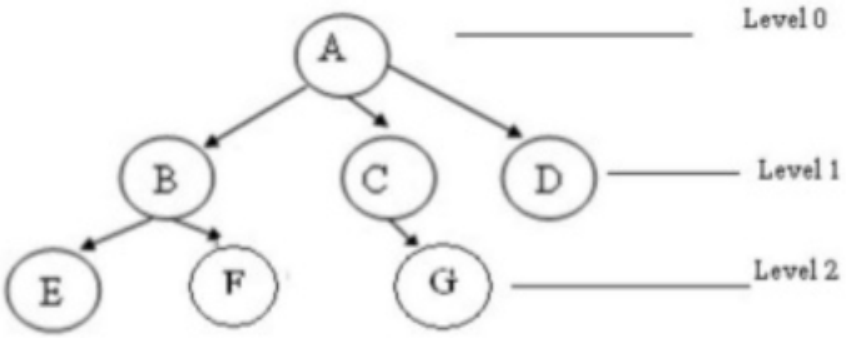
Trees

This is how a normal tree looks like

But for programmers, it's all different. It's inverted.

Terminology:-

- ➢ **Node :** Each element in the tree.
- ➢ **Edge :** Line connecting nodes.
- ➢ **Parent Node :** Immediate predecessor of a Node.
- ➢ **Child Node :** Immediate successor of a Node.
- ➢ **Root Node :** Node without parent.
- ➢ **Grand Parent Node :** Parent of Parent.
- ➢ **Level :** Distance of Node from root.
- ➢ **Siblings :** Children of the same parent Node.
- ➢ **Degree** : Maximum number of children a node can have.
- ➢ **Leaf Node** : Node without children.
- ➢ **Height/Depth** : Level of tree counted from root to the lowermost leaf node.
- ➢ **External Node** : All Leaf Nodes.
- ➢ **Internal Nodes** : All Non-Leaf Nodes.



- ✔ A is the root node
- ✔ B is the parent of E and F
- ✔ D is the sibling of B and C
- ✔ E and F are children of B
- ✔ E, F, G, D are external nodes or leaves
- ✔ A, B, C are internal nodes
- ✔ Depth of F is 2
- ✔ the height of tree is 2
- ✔ the degree of node A is 3
- ✔ The degree of tree is 3

Binary Trees :-

Tree with degree two is called a **Binary Tree.**

Class Representation of a tree :-

```
public class BTNode {
     int data;
     BTNode[] children = new BTNode[degree];
}
```

For a BinaryTree, the degree is 2, so, it can be represented as below :

```
public class BTNode {
     int data;
     BTNode[] children = new BTNode[2];
}
```

BinaryTree can be represented as below to understand easier :

```
public class BTNode{
     int data;
     Node left;
     Node right;
}
```