Experiment No :

Title:

A book consists of chapters, chapters consist of sections and sections consist of subsections. Construct a tree and print the nodes. Find the time and space requirements of your method.

Objectives:

1. To understand concept of tree data structure
2. To understand concept & features of object oriented programming.

Learning Objectives:

To understand concept of class

To understand concept & features of object oriented programming.

To understand concept of tree data structure.

Learning Outcome:

Define class for structures using Object Oriented features.

Analyze tree data structure.

Theory:

Introduction to Tree:

Definition:

A tree T is a set of nodes storing elements such that the nodes have a parent-child relationship that satisfies the following

• if T is not empty, T has a special tree called the root that has no parent

• each node v of T different than the root has a unique parent node w; each node with parent w is a child of w

Recursive definition

• T is either empty

• or consists of a node r (the root) and a possibly empty set of trees whose roots are the children of r

Tree is a widely-used data structure that emulates a tree structure with a set of linked nodes. The tree graphicaly is represented most commonly as on *Picture 1*. The circles are the nodes and the edges are the links between them.

Trees are usualy used to store and represent data in some hierarhical order. The data are stored in the nodes, from which the tree is consisted of.
A node may contain a value or a condition or represent a separate data structure or a tree of its own. Each node in a tree has zero or more child nodes, which are one level lower in the tree hierarchy (by convention, trees grow down, not up as they do in nature).
A node that has a child is called the child's parent node (or ancestor node, or superior). A node has at most one parent. A node that has no childs is called a leaf, and that node is of course at the bottommost level of the tree. The height of a node is the length of the longest path to a leaf from that node. The height of the root is the height of the tree. In other words, the "height" of tree is the "number of levels" in the tree. Or more formaly, the height of a tree is defined as follows:
1. The height of a tree with no elements is 0
2. The height of a tree with 1 element is 1
3. The height of a tree with > 1 element is equal to 1 + the height of its tallest subtree.

The depth of a node is the length of the path to its root (i.e., its root path). Every child node is always one level lower than his parent.
The topmost node in a tree is called the root node. Being the topmost node, the root node will not have parents. It is the node at which operations on the tree commonly begin (although some algorithms begin with the leaf nodes and work up ending at the root). All other nodes can be reached from it by following edges or links. (In the formal definition, a path from a root to a node, for each different node is always unique).
In diagrams, it is typically drawn at the top.
In some trees, such as heaps, the root node has special properties.

A subtree is a portion of a tree data structure that can be viewed as a complete tree in itself. Any node in a tree T, together with all the nodes below his height, that are reachable from the node, comprise a subtree of T. The subtree corresponding to the root node is the entire tree; the subtree corresponding to any other node is called a proper subtree (in analogy to the term proper subset).
Every node in a tree can be seen as the root node of the subtree rooted at that node.

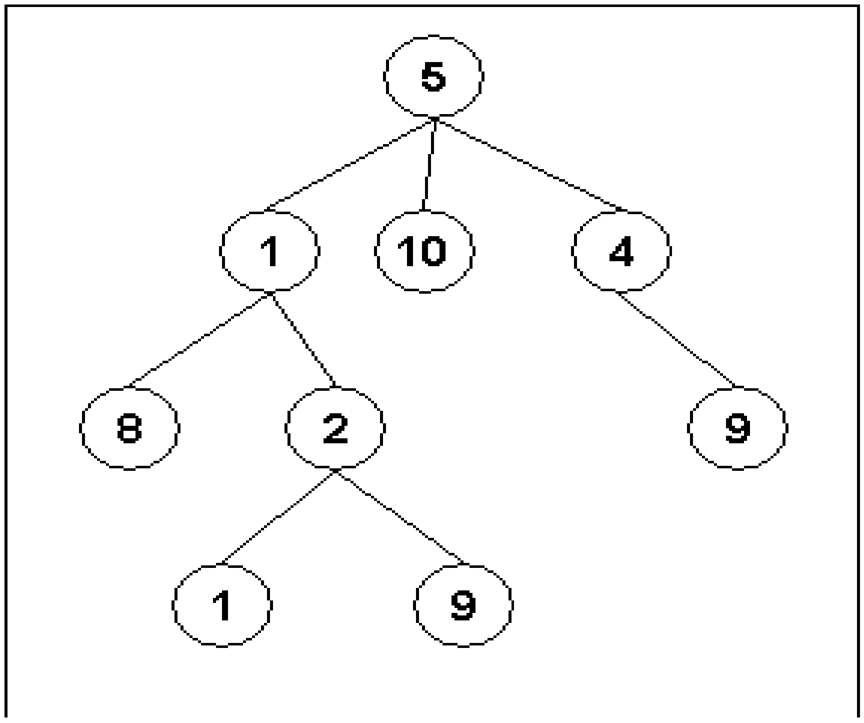


Fig1. An example of a tree

An internal node or inner node is any node of a tree that has child nodes and is thus not a leaf node.
There are two basic types of trees. In an unordered tree, a tree is a tree in a purely structural sense — that is to say, given a node, there is no order for the children of that node. A tree on which an order is imposed — for example, by assigning different natural numbers to each child of each node — is called an ordered tree, and data structures built on them are called ordered tree data structures. Ordered trees are by far the most common form of tree data structure. Binary search trees are one kind of ordered tree.

Important Terms

Following are the important terms with respect to tree.

Path − Path refers to the sequence of nodes along the edges of a tree.

Root − The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.

Parent − Any node except the root node has one edge upward to a node called parent.

Child − The node below a given node connected by its edge downward is called its child node.

Leaf − The node which does not have any child node is called the leaf node.

Subtree − Subtree represents the descendants of a node.

Visiting − Visiting refers to checking the value of a node when control is on the node.

Traversing − Traversing means passing through nodes in a specific order.

Levels − Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.

keys − Key represents a value of a node based on which a search operation is to be carried out for a node.

Advantages of trees

Trees are so useful and frequently used, because they have some very serious advantages:

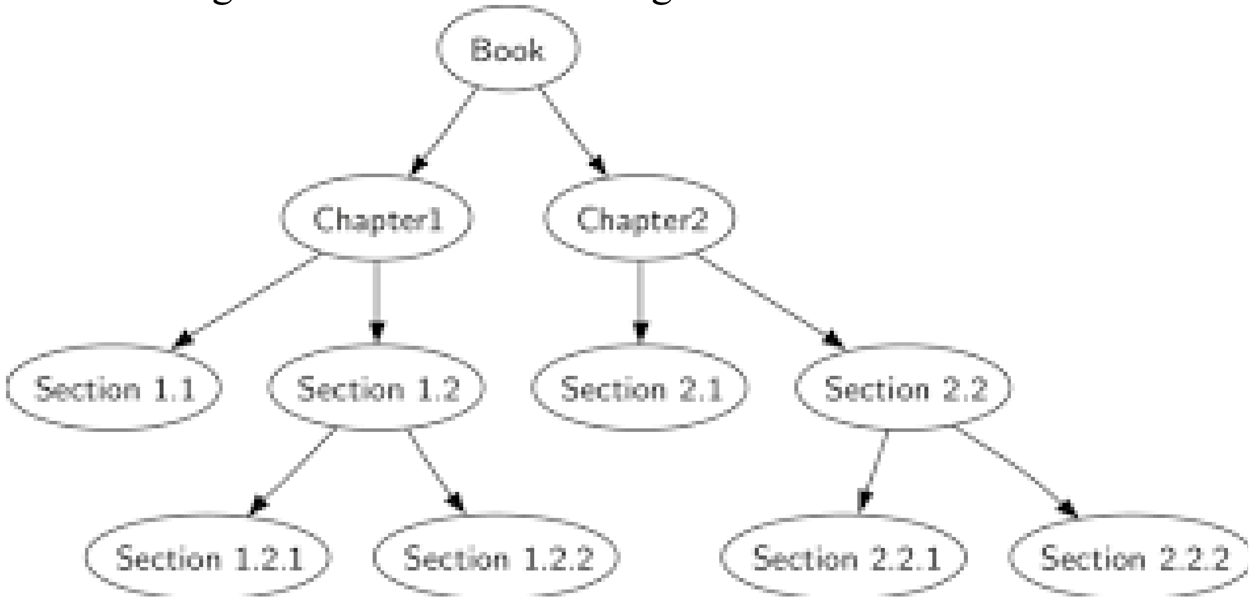
Trees reflect structural relationships in the data

Trees are used to represent hierarchies

Trees provide an efficient insertion and searching

Trees are very flexible data, allowing to move subtrees around with minumum effort

For this assignment we are considering the tree as follows.



Software Required: g++ / gcc compiler- / 64 bit Fedora, eclipse IDE
Input: Book name & its number of sections and subsections along with name.

Output: Formation of tree structure for book and its sections.

Conclusion: Write in your own words.