Chapter 10 introduces trees.

This presentation illustrates the simplest kind of trees: Complete Binary Trees.
Binary Trees

- A binary tree has **nodes**, similar to nodes in a linked list structure.
- **Data** of one sort or another may be stored at each node.
- But it is the **connections** between the nodes which characterize a binary tree.
Binary Trees

- A binary tree has **nodes**, similar to nodes in a linked list structure.
- **Data** of one sort or another may be stored at each node.
- But it is the **connections** between the nodes which characterize a binary tree.

An example can illustrate how the connections work.
A Binary Tree of States

In this example, the data contained at each node is one of the 50 states.
A Binary Tree of States

Each tree has a special node called its root, usually drawn at the top.
A Binary Tree of States

Each tree has a special node called its root, usually drawn at the top.

The example tree has Washington as its root.
A Binary Tree of States

Each node is permitted to have two links to other nodes, called the **left child** and the **right child**.
A Binary Tree of States

Each node is permitted to have two links to other nodes, called the left child and the right child.
A Binary Tree of States

Children are usually drawn below a node.

The left child of Washington is Arkansas.

The right child of Washington is Colorado.
A Binary Tree of States

Some nodes have only one child.

Arkansas has a left child, but no right child.
A Quiz

Some nodes have only one child.

Which node has only a right child?
A Quiz

Some nodes have only one child.

Florida has only a right child.
A Binary Tree of States

A node with no children is called a leaf.
A Binary Tree of States

Each node is called the **parent** of its children.

Washington is the parent of Arkansas and Colorado.
A Binary Tree of States

Two rules about parents:

1. The root has no parent.
2. Every other node has exactly one parent.
Two nodes with the same parent are called siblings.

Arkansas and Colorado are siblings.
Complete Binary Trees

A **complete** binary tree is a special kind of binary tree which will be useful to us.
Complete Binary Trees

A complete binary tree is a special kind of binary tree which will be useful to us.

When a complete binary tree is built, its first node must be the root.
Complete Binary Trees

The second node of a complete binary tree is always the left child of the root...
Complete Binary Trees

The second node of a complete binary tree is always the left child of the root...

... and the third node is always the right child of the root.
Complete Binary Trees

The next nodes must always fill the next level from *left to right*. 
Complete Binary Trees

The next nodes must always fill the next level from **left to right**.
Complete Binary Trees

The next nodes must always fill the next level from left to right.
Complete Binary Trees

The next nodes must always fill the next level from **left to right**.
Complete Binary Trees

The next nodes must always fill the next level from **left to right**.
Complete Binary Trees

The next nodes must always fill the next level from **left to right**.
Is This Complete?
Is This Complete?
Is This Complete?
Is This Complete?

Yes!

✔ It is called the empty tree, and it has no nodes, not even a root.
Implementing a Complete Binary Tree

- We will store the date from the nodes in a partially-filled array.

An integer to keep track of how many nodes are in the tree

An array of data

We don't care what's in this part of the array.
Implementing a Complete Binary Tree

- We will store the date from the nodes in a partially-filled array.

3

An integer to keep track of how many nodes are in the tree

Read Section 10.2 to see details of how the entries are stored.
Binary trees contain nodes.
Each node may have a left child and a right child.
If you start from any node and move upward, you will eventually reach the root.
Every node except the root has one parent. The root has no parent.
Complete binary trees require the nodes to fill in each level from left-to-right before starting the next level.
Presentation copyright 1997 Addison Wesley Longman,
For use with Data Structures and Other Objects Using C++
by Michael Main and Walter Savitch.

Some artwork in the presentation is used with permission from Presentation Task Force
(copyright New Vision Technologies Inc) and Corel Gallery Clipart Catalog (copyright
Corel Corporation, 3G Graphics Inc, Archive Arts, Cartesia Software, Image Club

Students and instructors who use Data Structures and Other Objects Using C++ are welcome
to use this presentation however they see fit, so long as this copyright notice remains
intact.

THE END