Lab 11
Preview

6 April 2017

In this lab, you'll write some code that builds, and then traverses, binary trees. For simplicity, we'll write trees that only hold characters.

Tree nodes

First, create a struct `BinaryNode`, capable of representing any of the nodes in a binary tree. It will have three instance variables: a **`char`**, holding the value that is stored at a particular node, and two pointers to `BinaryNode` (one to the left child, if any, and one to the right child, if any).

By now you should be getting comfortable with writing your own structs and classes, so I won't recap those instructions here; look back at previous labs to help you remember how.

Examples

In a notebook, draw out the following three trees:

- `emptyTree`, which is simply set to **`nullptr`**
- `simple`, which points to a node containing `'Q'` whose left child contains `'X'` and right child contains `'Z'` (and no further descendants)
- `tree5`, pointing to a node that is the root of a small tree that contains the five letters `'A'` through `'E'` and is relatively balanced (i.e. not just a line)

Near each tree, write out the C++ expression you will use to actually construct the corresponding tree. (You'll have these in a `.u` file as well, eventually, but putting them in your notebook makes it easier for me *and* you to refer back to them quickly.)