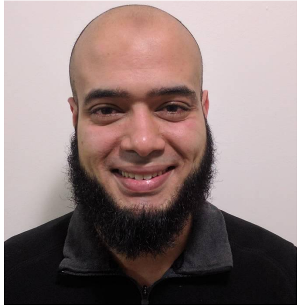
Data Structures

Binary Tree Creation

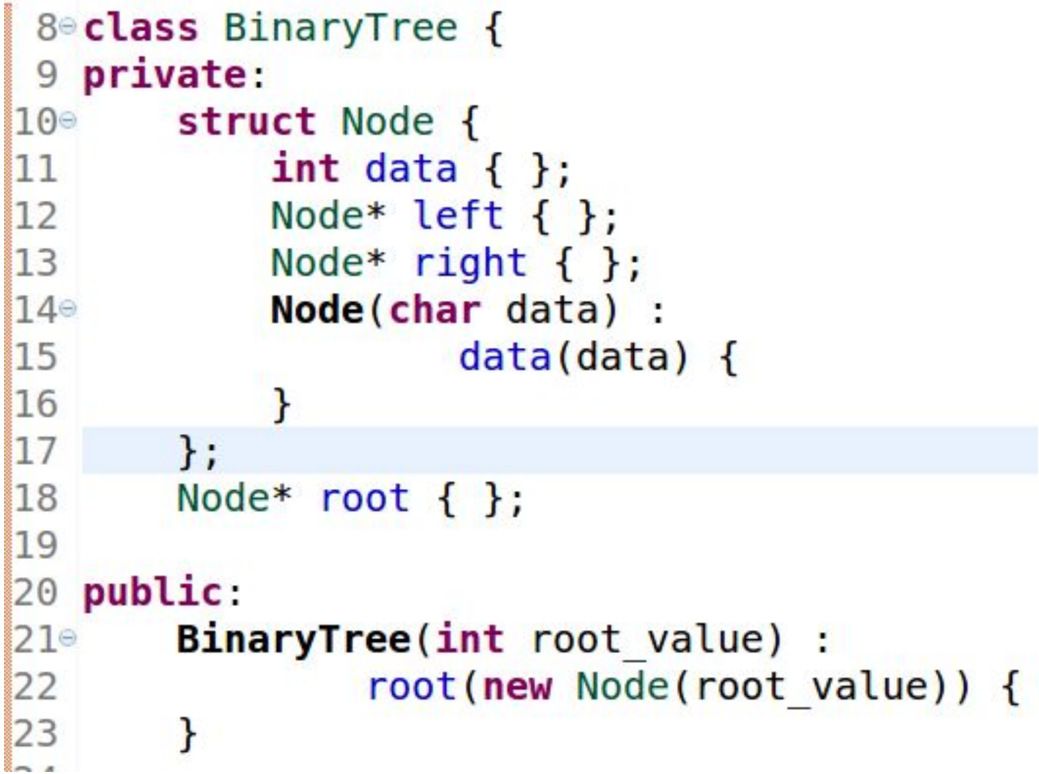
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



So far

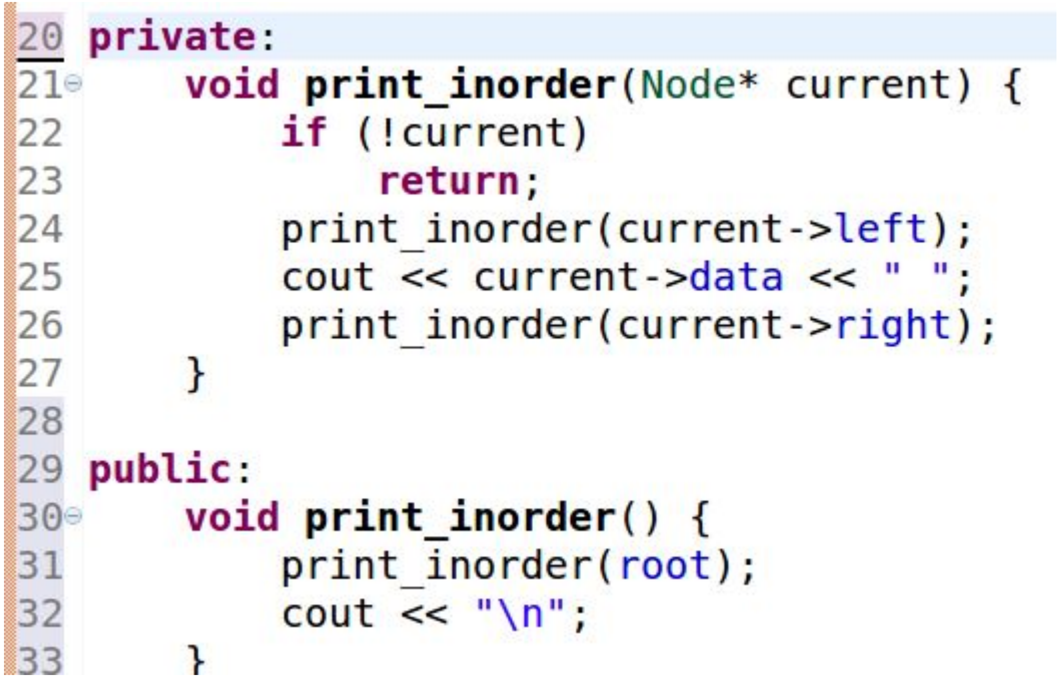
- We learned manual creation
- 3 Traversal ways
- Let’s create tree structure
- Then decide how to add edges
- We create tree class, that has a root node



```
class BinaryTree {
private:
    struct Node {
        int data { };
        Node* left { };
        Node* right { };
        Node(char data) :
                data(data) {
        }
    };
    Node* root { };

public:
    BinaryTree(int root_value) :
            root(new Node(root_value)) {
    }
```

Adding in-order traversal

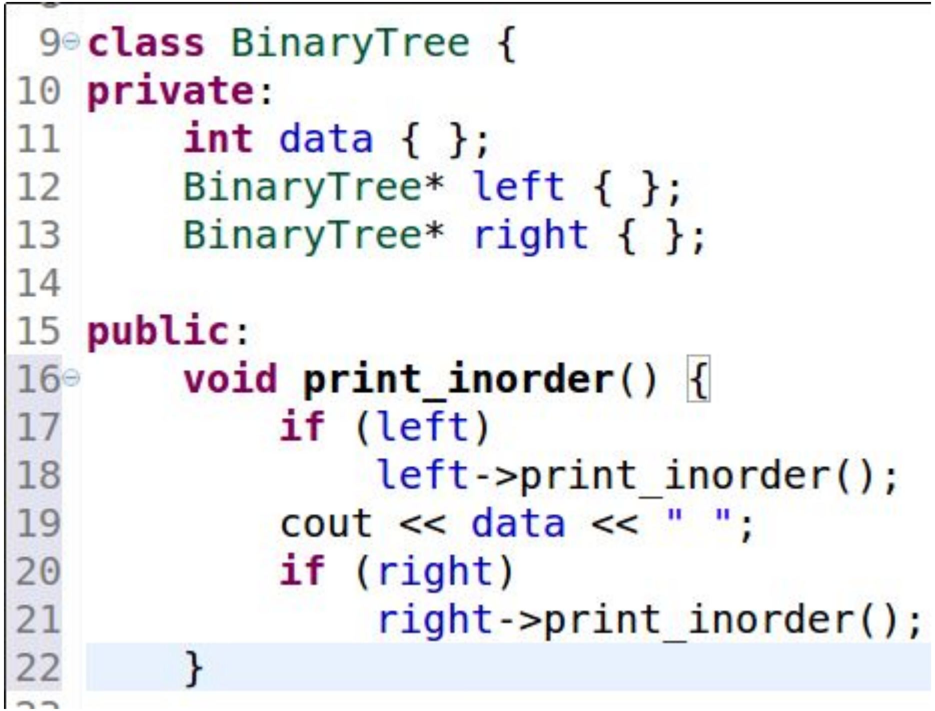
- Let’s move the inorder inside the class
- Some extra handling to let outsiders call it
- Also print line after printing



```
private:
    void print_inorder(Node* current) {
        if (!current)
            return;
        print_inorder(current->left);
        cout << current->data << " ";
        print_inorder(current->right);
    }

public:
    void print_inorder() {
        print_inorder(root);
        cout << "\n";
    }
```

Another approach

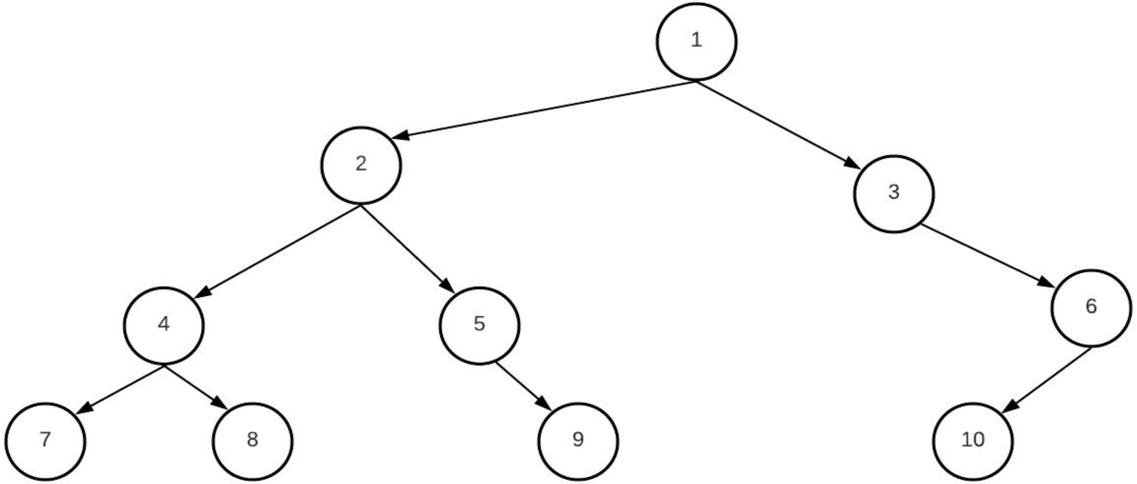
- I prefer another way than the mainstream style.
- Now, we don’t have to do this 2 functions style or Node* as parameter
  - Cons: checks for if(left)



```
class BinaryTree {
private:
    int data { };
    BinaryTree* left { };
    BinaryTree* right { };

public:
    void print_inorder() {
        if (left)
            left->print_inorder();
        cout << data << " ";
        if (right)
            right->print_inorder();
    }
```

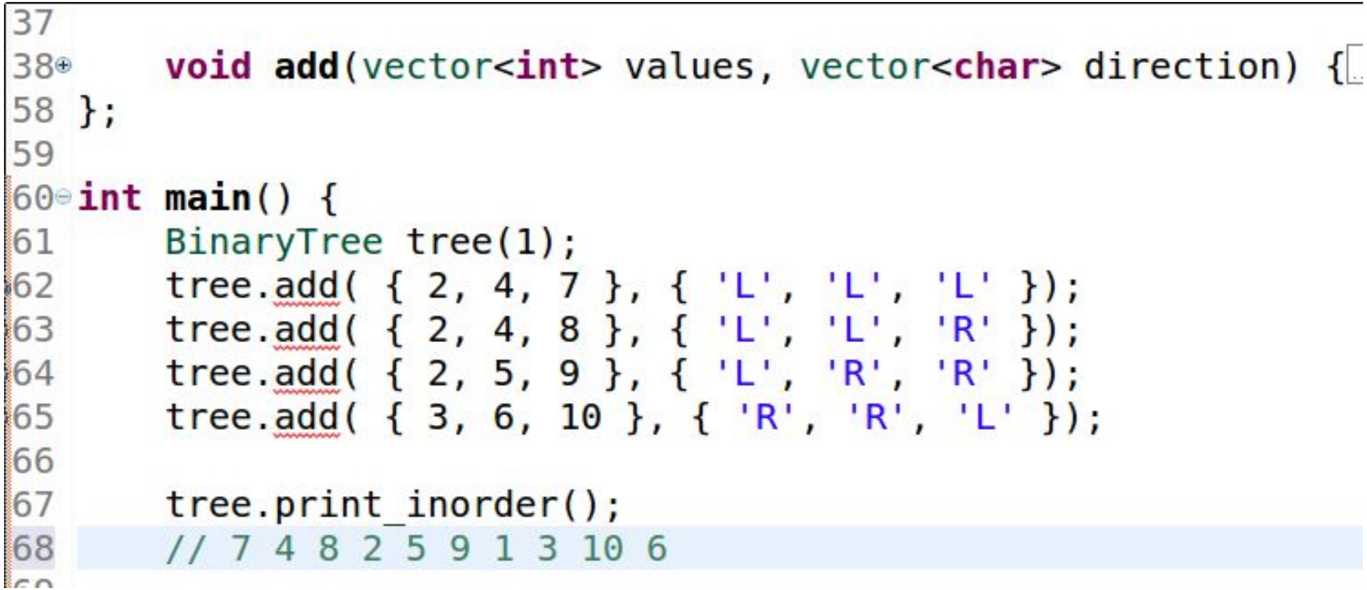
How to construct such a tree?

- Here is 1 way: for each leaf node,
- **Add path nodes**
  - 1 ⇒ 2 ⇒ 4 ⇒ 7
  - 1 ⇒ 2 ⇒ 4 ⇒ 8
  - 1 ⇒ 2 ⇒ 5 ⇒ 9
  - 1 ⇒ 3 ⇒ 6 ⇒ 10
- **Add path directions**
  - LLL
  - LLR
  - LRR
  - RRL



Construction

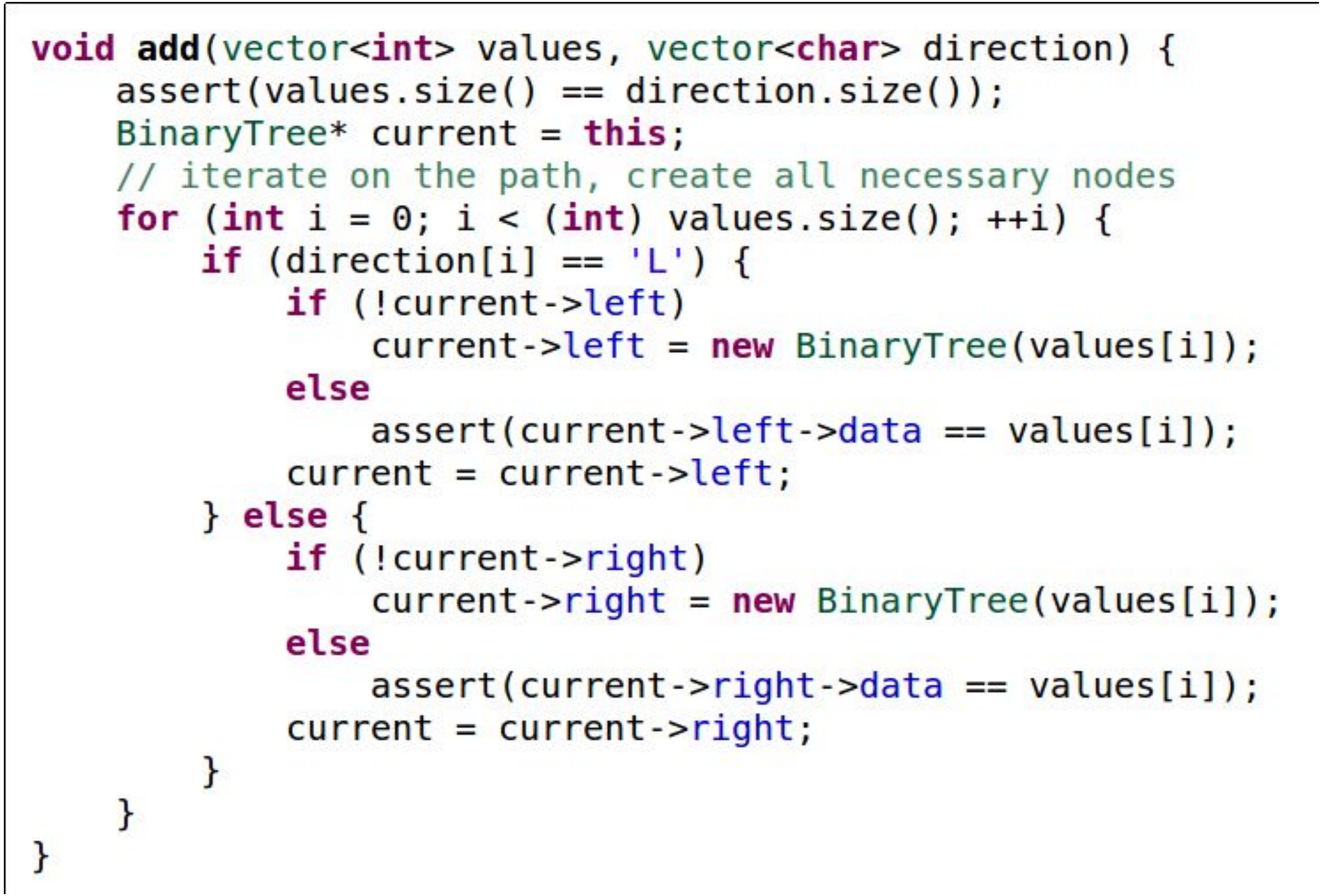
- It is also good to verify paths doesn’t conflict!
  - All paths passing with a created node, it must have the same old value



```

    void add(vector<int> values, vector<char> direction) {
};

int main() {
    BinaryTree tree(1);
    tree.add( { 2, 4, 7 }, { 'L', 'L', 'L' });
    tree.add( { 2, 4, 8 }, { 'L', 'L', 'R' });
    tree.add( { 2, 5, 9 }, { 'L', 'R', 'R' });
    tree.add( { 3, 6, 10 }, { 'R', 'R', 'L' });

    tree.print_inorder();
    // 7 4 8 2 5 9 1 3 10 6
```

Construction

- We already set root value
- Each path is for remaining children



```
void add(vector<int> values, vector<char> direction) {
    assert(values.size() == direction.size());
    BinaryTree* current = this;
    // iterate on the path, create all necessary nodes
    for (int i = 0; i < (int) values.size(); ++i) {
        if (direction[i] == 'L') {
            if (!current->left)
                current->left = new BinaryTree(values[i]);
            else
                assert(current->left->data == values[i]);
            current = current->left;
        } else {
            if (!current->right)
                current->right = new BinaryTree(values[i]);
            else
                assert(current->right->data == values[i]);
            current = current->right;
        }
    }
}
```

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”