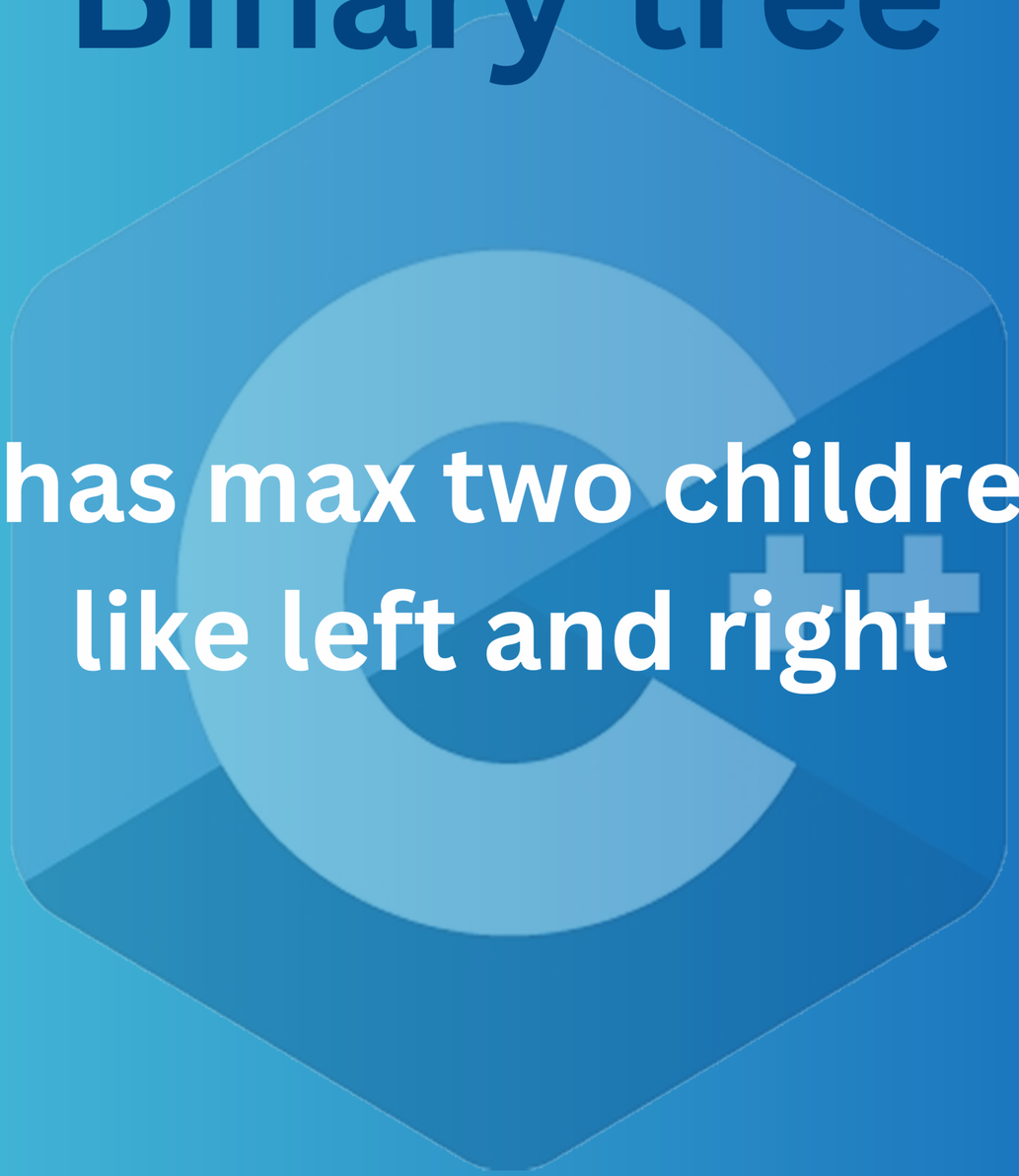
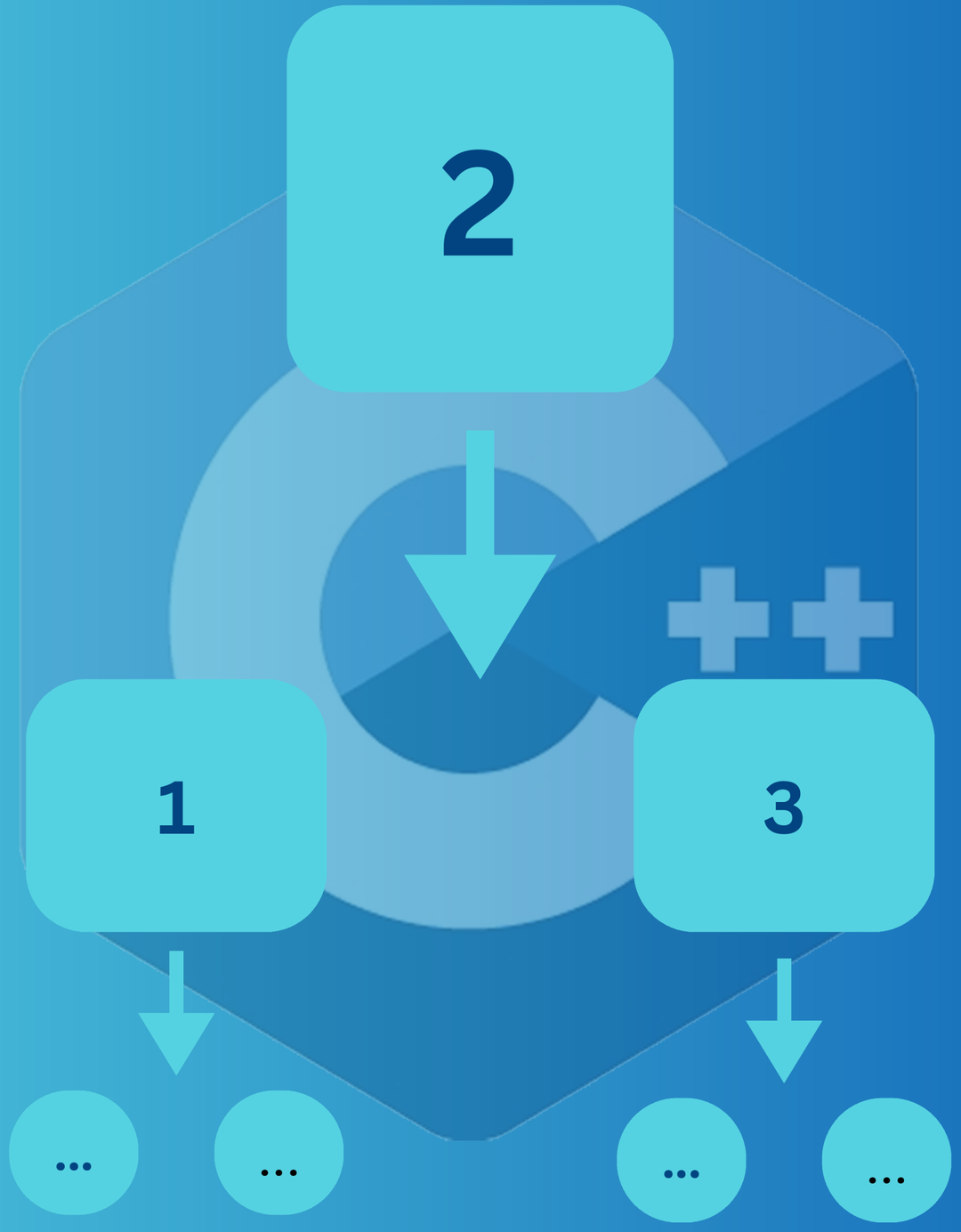
Tree and Binary tree

Tree is a Hierarchical data structure with nodes like Root, branches, leaves that represent connections.

Binary tree



Each node has max two children or leaves like left and right


2
1
3
...
...
...
...

tree.cpp U


```
53  };
54
55  int main (){
56      Tree t;
57
58      t.add(2);
59      t.add(1);
60      t.add(3);
61
62      cout << t.root -> data << endl;
63      cout << t.root -> left -> data << endl;
64      cout << t.root -> right -> data<< endl;
65  }
```

PROBLEMS DEBUG CONSOLE OUTPUT PORTS SERIAL MONITOR COMMENTS TERMINAL

powershell - code

```
PS D:\Documents\GitHub\Introduction-to-Data-Structure\code> g++ tree.cpp -o tree
PS D:\Documents\GitHub\Introduction-to-Data-Structure\code> g++ tree.cpp -o tree
PS D:\Documents\GitHub\Introduction-to-Data-Structure\code> ./tree.exe
2
1
3
PS D:\Documents\GitHub\Introduction-to-Data-Structure\code>
```

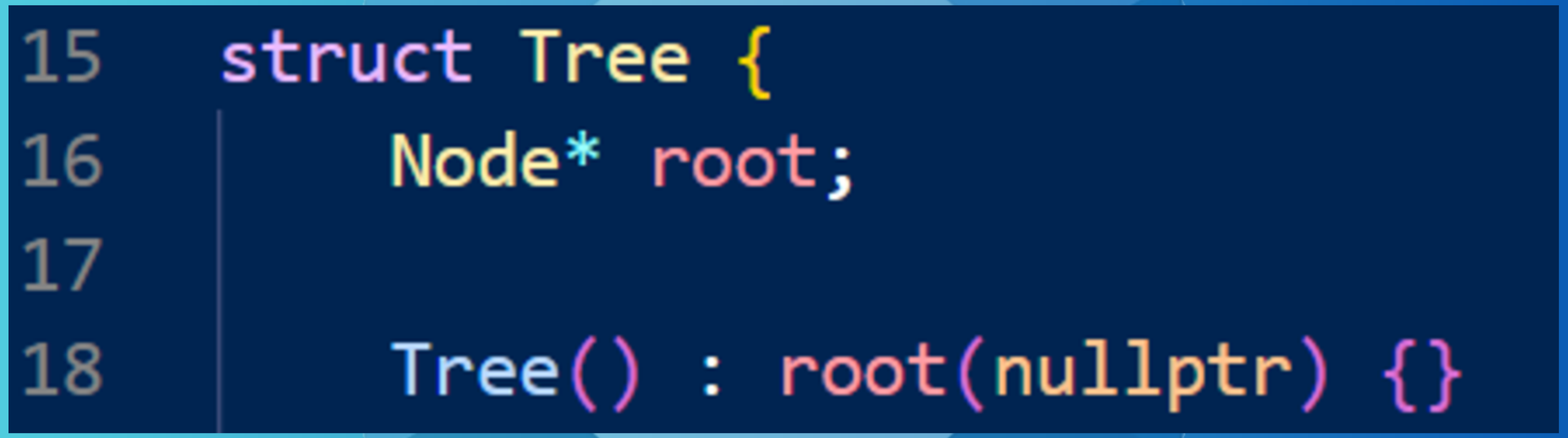
Code

```cpp
// tree stores elements in a tree like format with root and leaves (left and right)

#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* left;
    Node* right;

    //constructor
    Node(int v) : data(v), left(nullptr), right(nullptr) {}
};
```


15 struct Tree {
16 Node* root;
17
18 Tree() : root(nullptr) {}

```
20      // add
21      void add(int v){
22          Node* n = new Node(v);
23          if (root == nullptr){
24              root = n;
25          } else {
26              Node* t = root;
27              while (true) {
28                  // if value is < data in that root
29                  if (v < t -> data){
30                      // if roots left leave ( node ) is empty add that to that
31                      if (t -> left == nullptr){
32                          t -> left = n;
33                          break;
34                      } else {
35                          // else if left is not empty, root become the left node and it goes over
36                          t = t -> left;
37                      }
38                  }
```

```
39                  // if value is > data in that root
40                  if (v > t -> data){
41                      // if roots right leave ( node ) is empty add that to that
42                      if (t -> right == nullptr){
43                          t -> right = n;
44                          break;
45                      } else {
46                          // else if right is not empty, root become the right node and it goes over
47                          t = t -> right;
48                      }
49                  }
50              }
51          }
52          }
53  };
```

Next Video