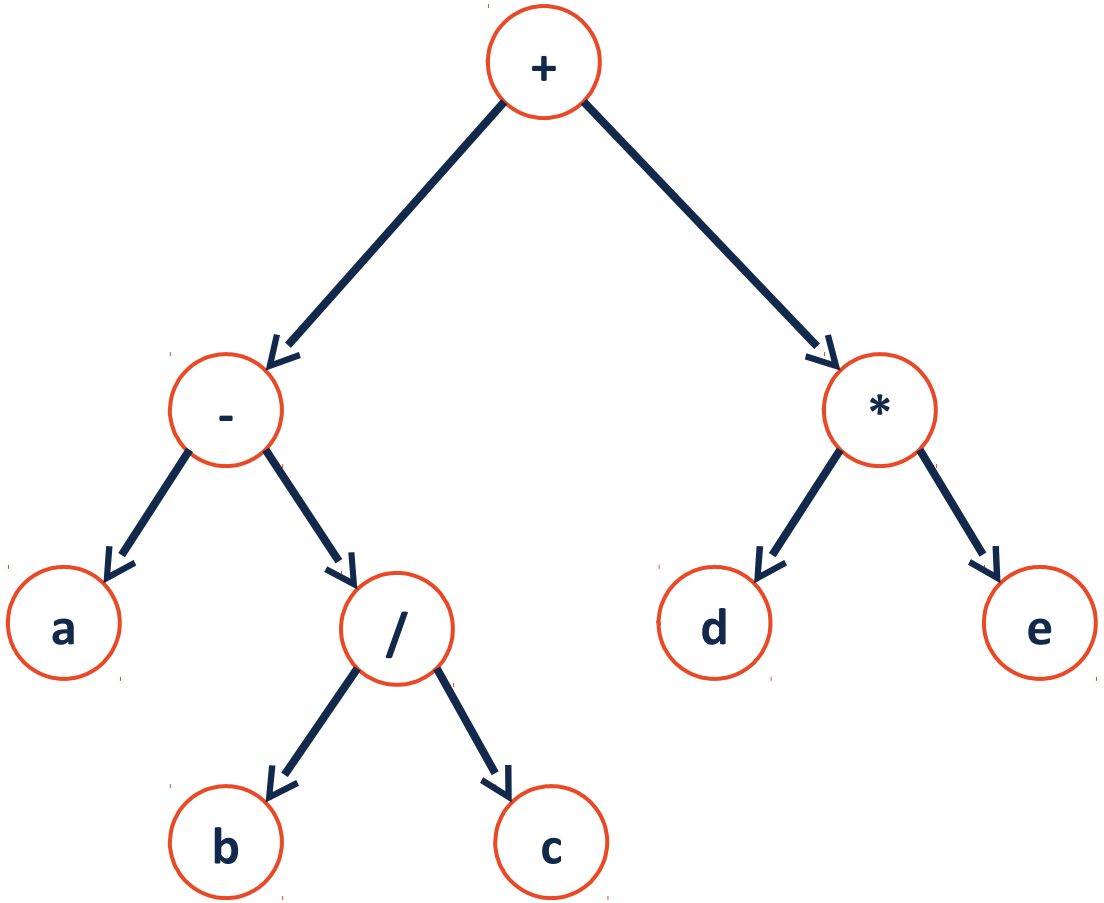
CS 400

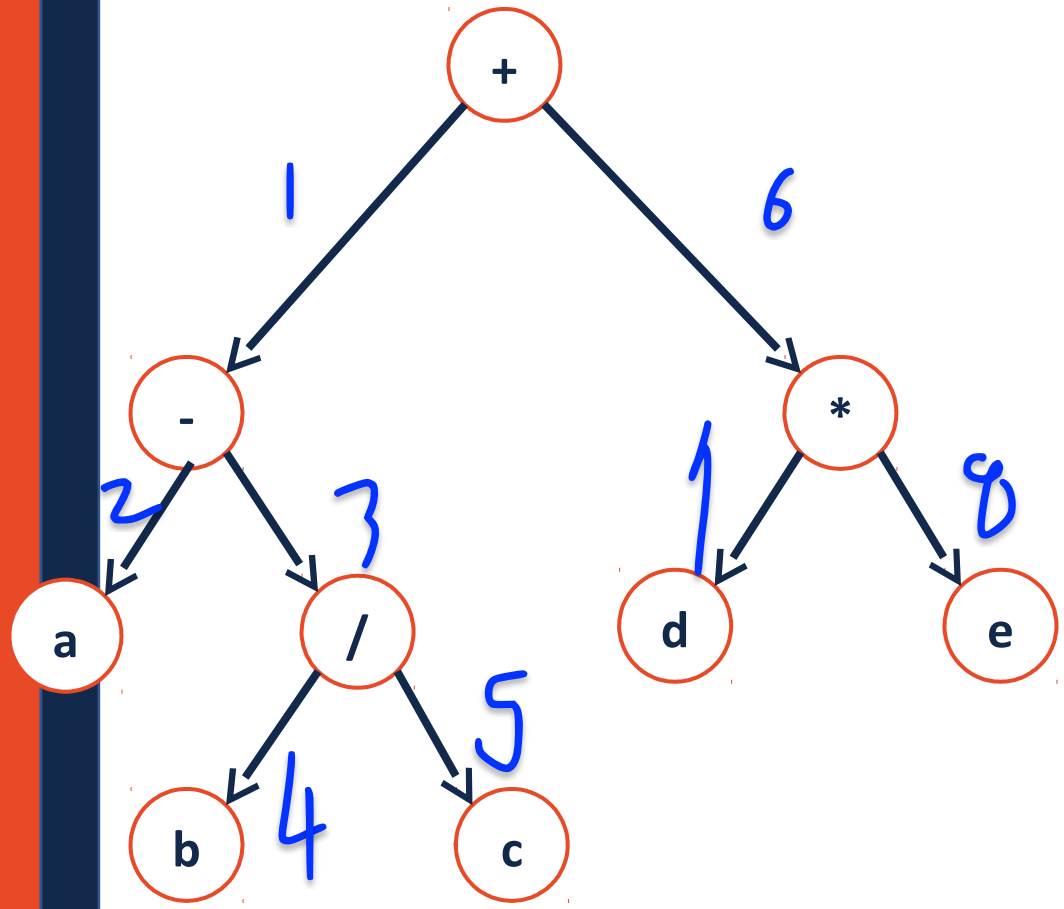
Tree Traversal

EXTRA-Tree-Traversal

Traversals



Traversals



Order : Shout Left Right

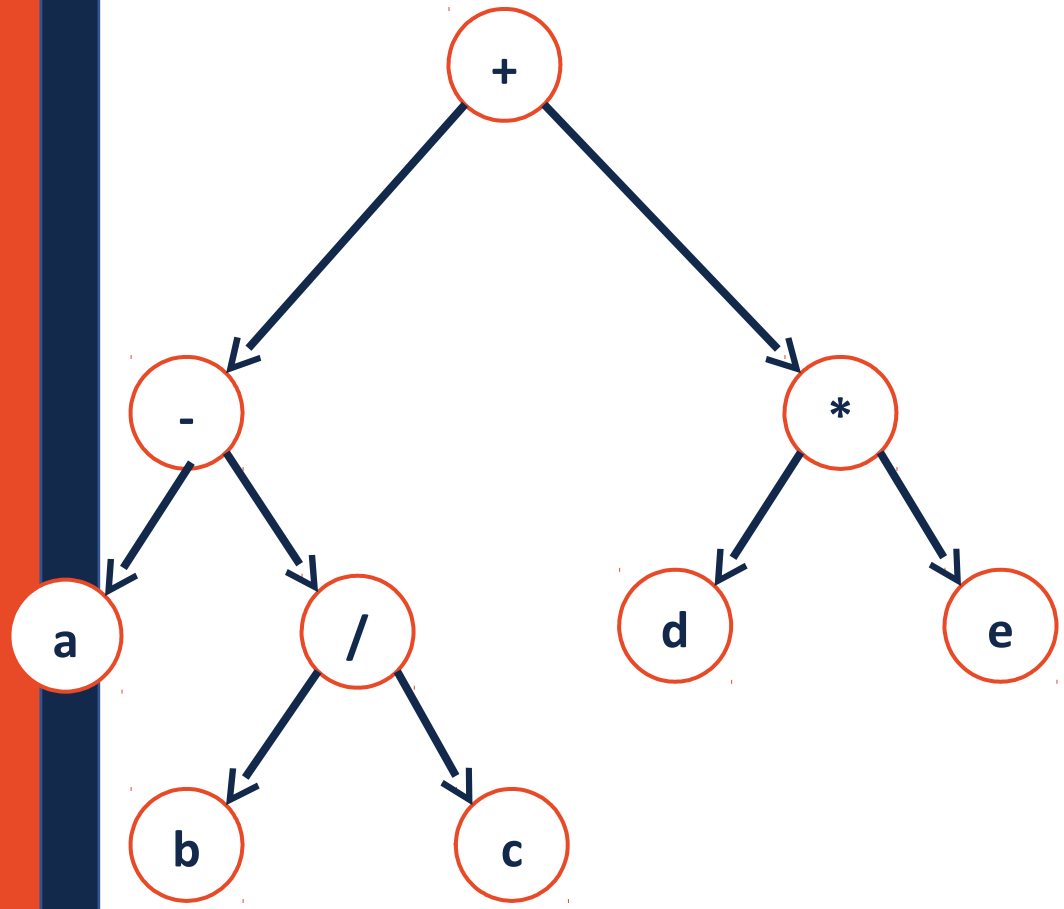
< Preorder Traversal >

```
49  template<class T>
50  void BinaryTree<T>::__Order(TreeNode * cur)
51  {
52
53
54
55
56
57
58  }
```

+ - a / b c * d e

A traversal needs to visit every node in our tree exactly once and do something with that data.

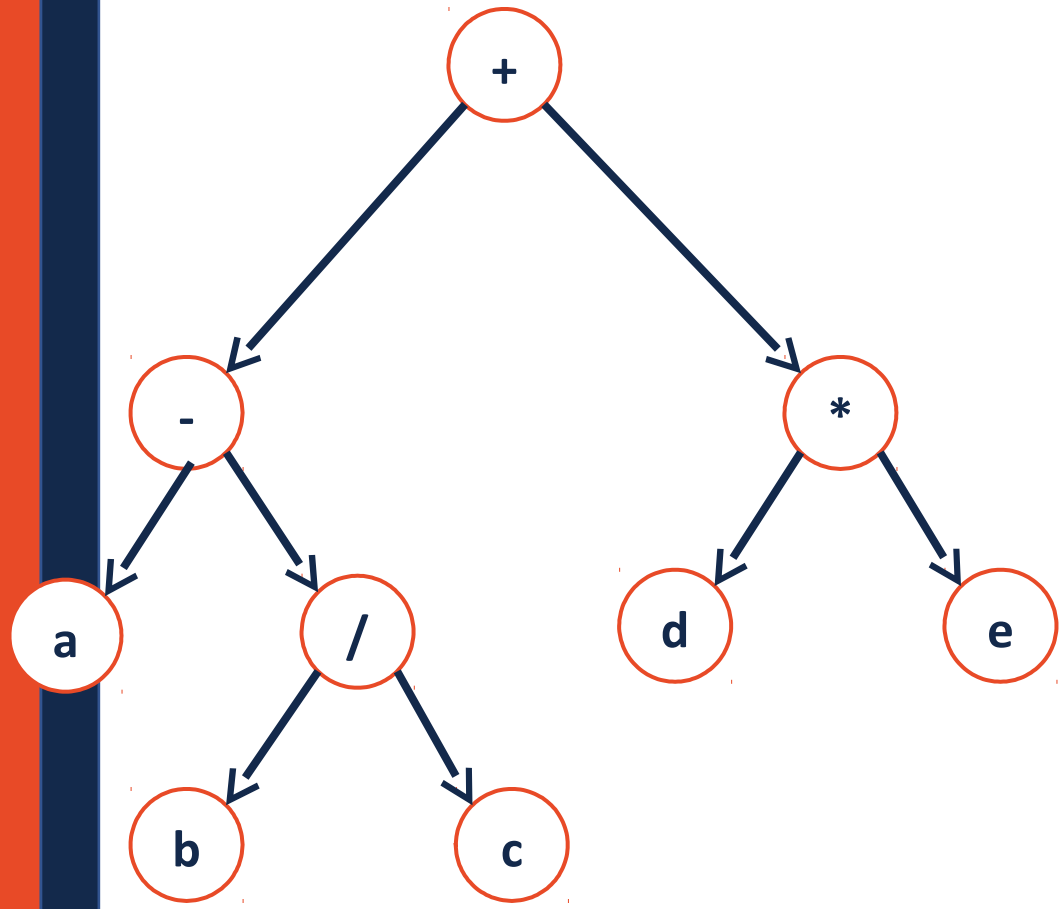
Traversals



왼쪽노드, 오른쪽 노드 모두 같은 함수를 재귀호출

```
49  template<class T>
50  void BinaryTree<T>::___Order(TreeNode * cur) {
51     if (cur != NULL) {
52        ____________________________;
53        ___Order(cur->left);
54        ____________________________;
55        ___Order(cur->right);
56        ____________________________;
57     }
58  }
```

Traversals



Order : Left Shout Right
< Inorder Traversal >

```
49  template<class T>
50  void BinaryTree<T>::___Order(TreeNode * cur) {
51     if (cur != NULL) {
52        ____________________________;
53        ___Order(cur->left);
54        ____________________________;
55        ___Order(cur->right);
56        ____________________________;
57     }
58  }
```

a - b / c + d * e

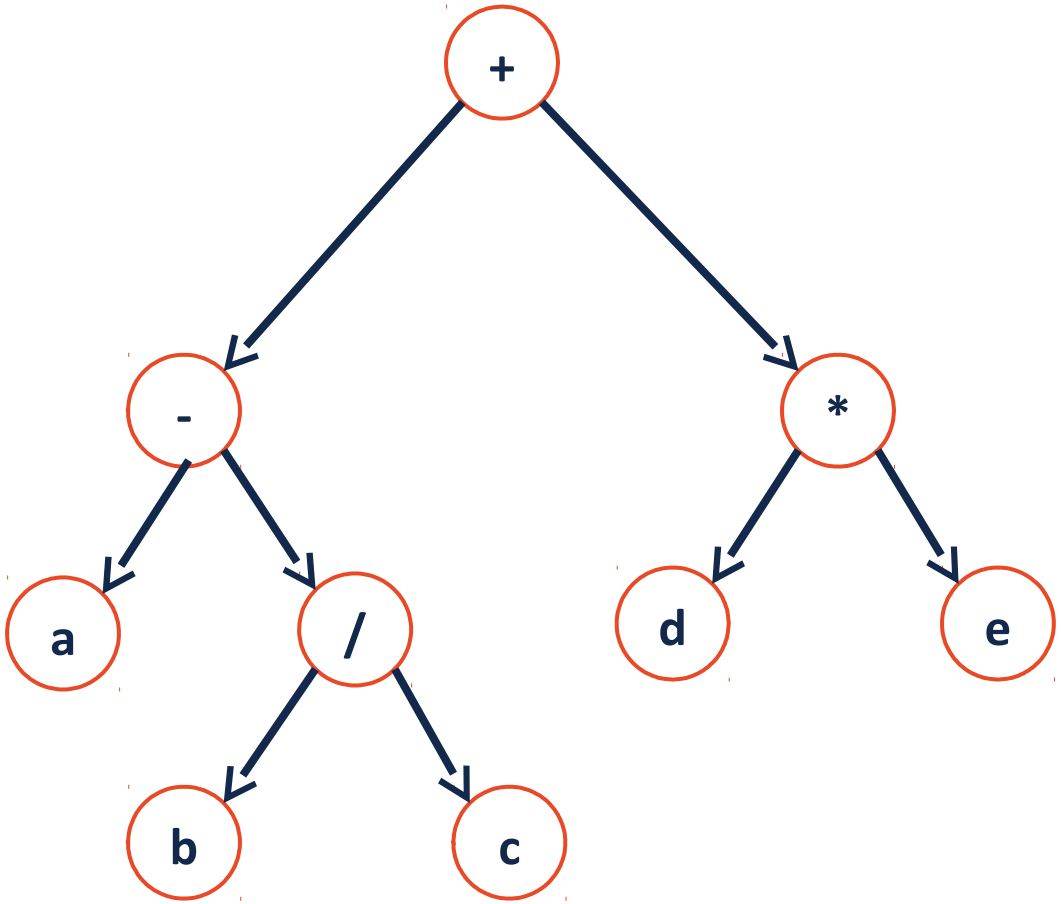
< Post-order Traversal >

Order : Left Right Shout

a b c / - d e *

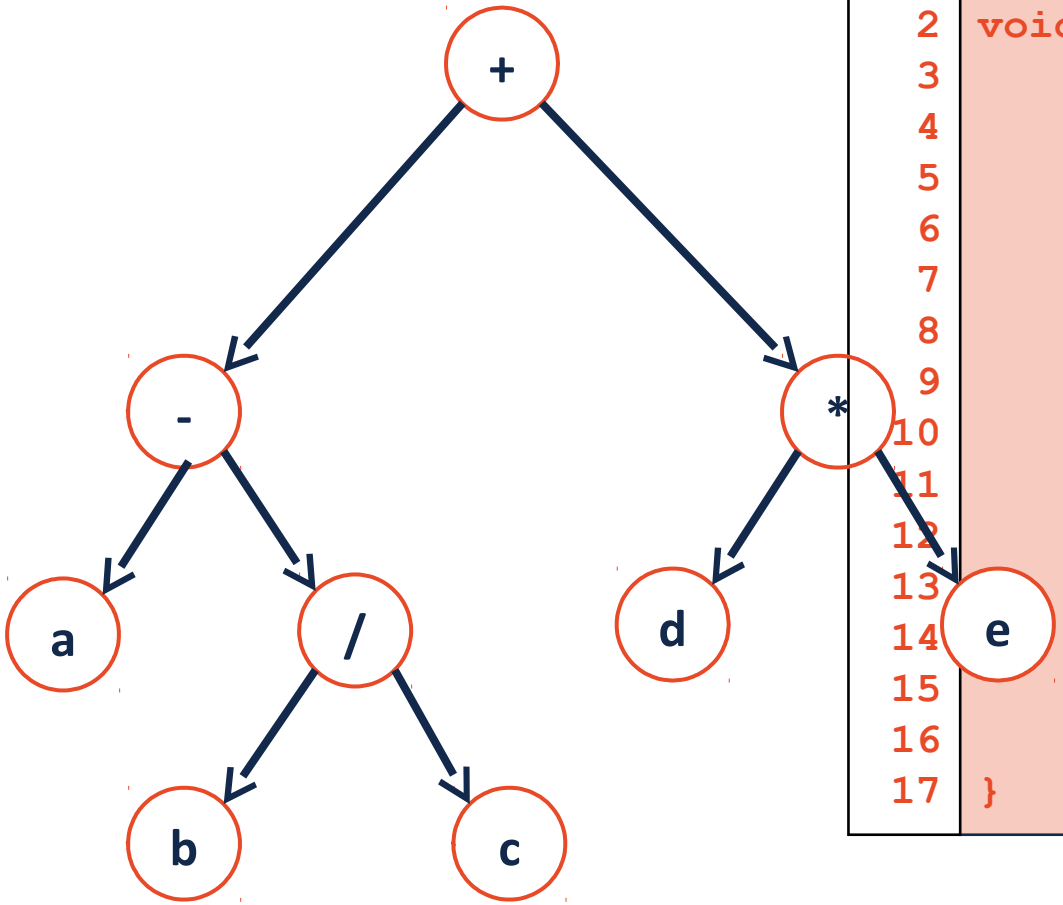
세 traversals는 shout의 위치가 어디냐만 다름

A Different Type of Traversal



Level order traversal : 레벨별로 가로로 횡단
+ - * a / d e b c

A Different Type of Traversal




```
1  template<class T>
2  void BinaryTree<T>::levelOrder(TreeNode * root) {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17 }
```

Traversal vs. Search

Traversal

A traversal needs to visit every node in our tree exactly once and do something with that data.

Search

A search allows us to discover a particular node throughout the tree.
We may not visit all nodes to search a specific node.
We quit search after we find what we want.