Lecture No.18
Expression Trees

CC-213 Data Structures
Department of Computer Science
University of the Punjab

Slides modified very slightly from the late Dr. Sohail Aslam's lectures at VU

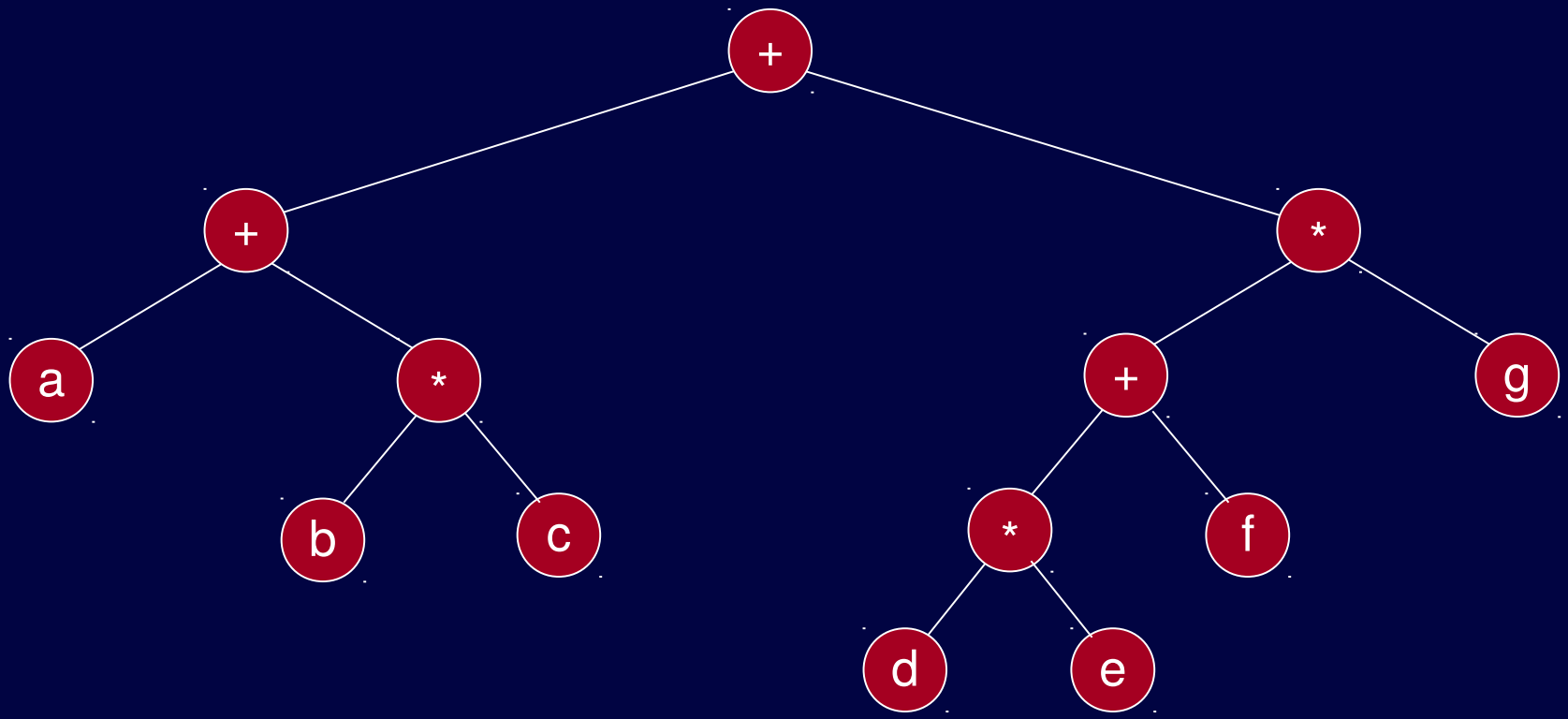
Lecture No.25

Data Structures

Dr. Sohail Aslam

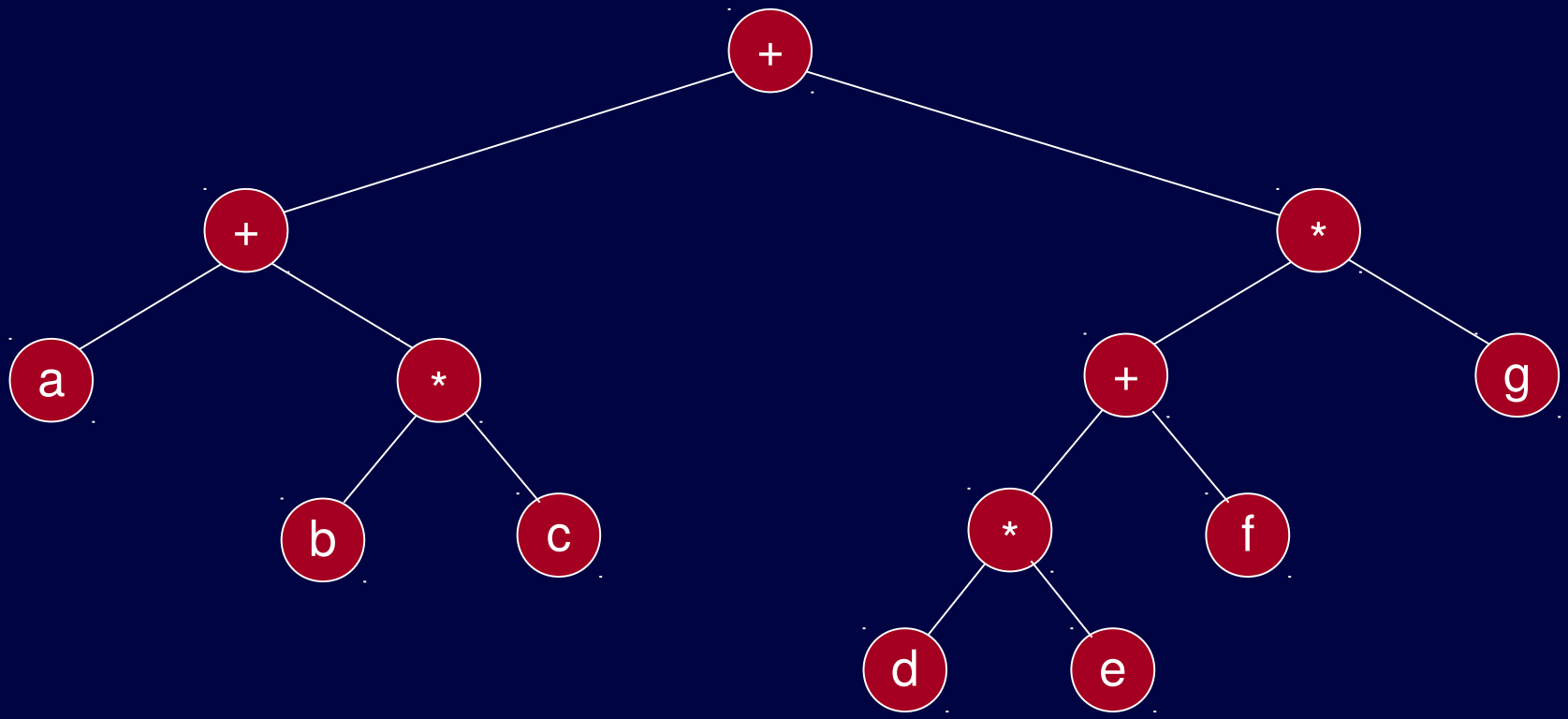
Expression Tree

- The inner nodes contain operators while leaf nodes contain operands.



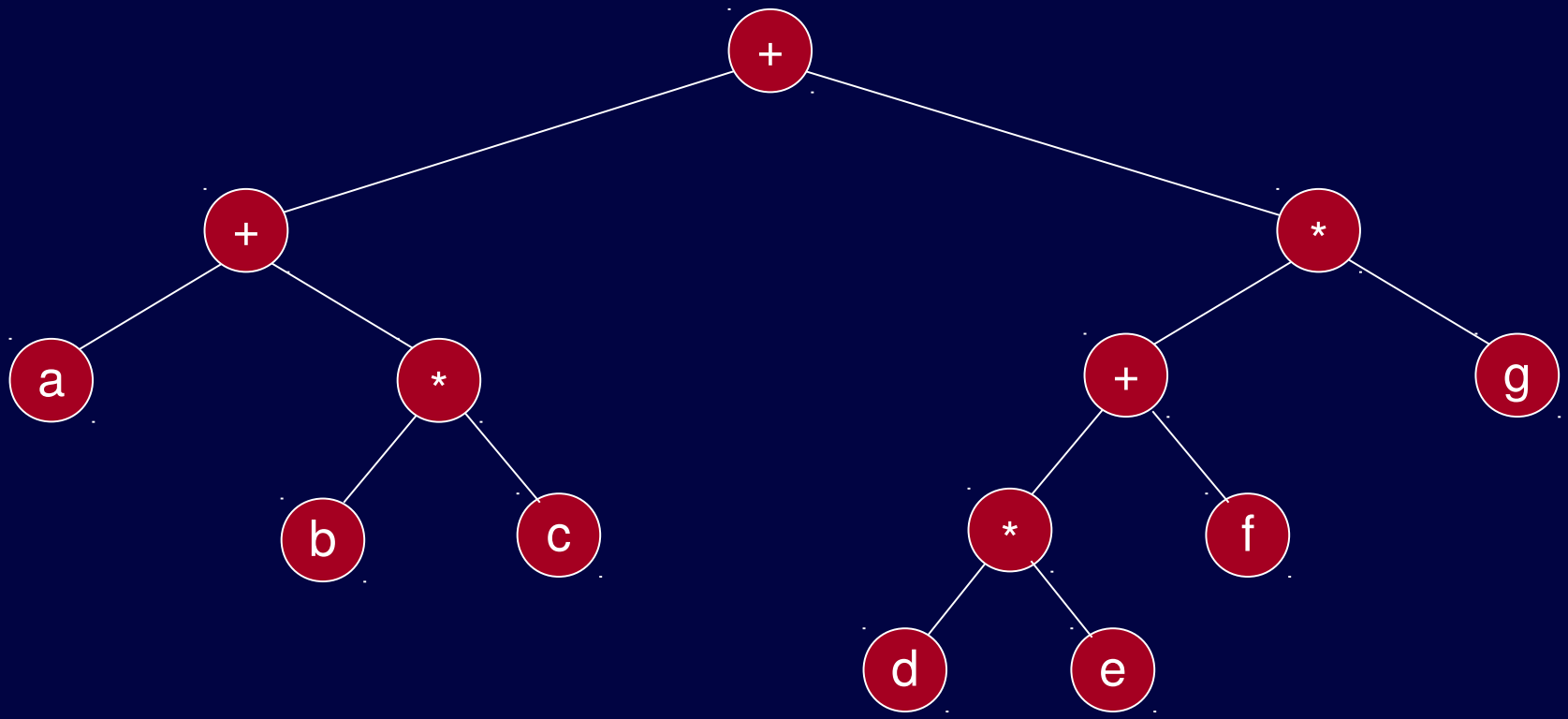
Expression Tree

- The tree is binary because the operators are binary.



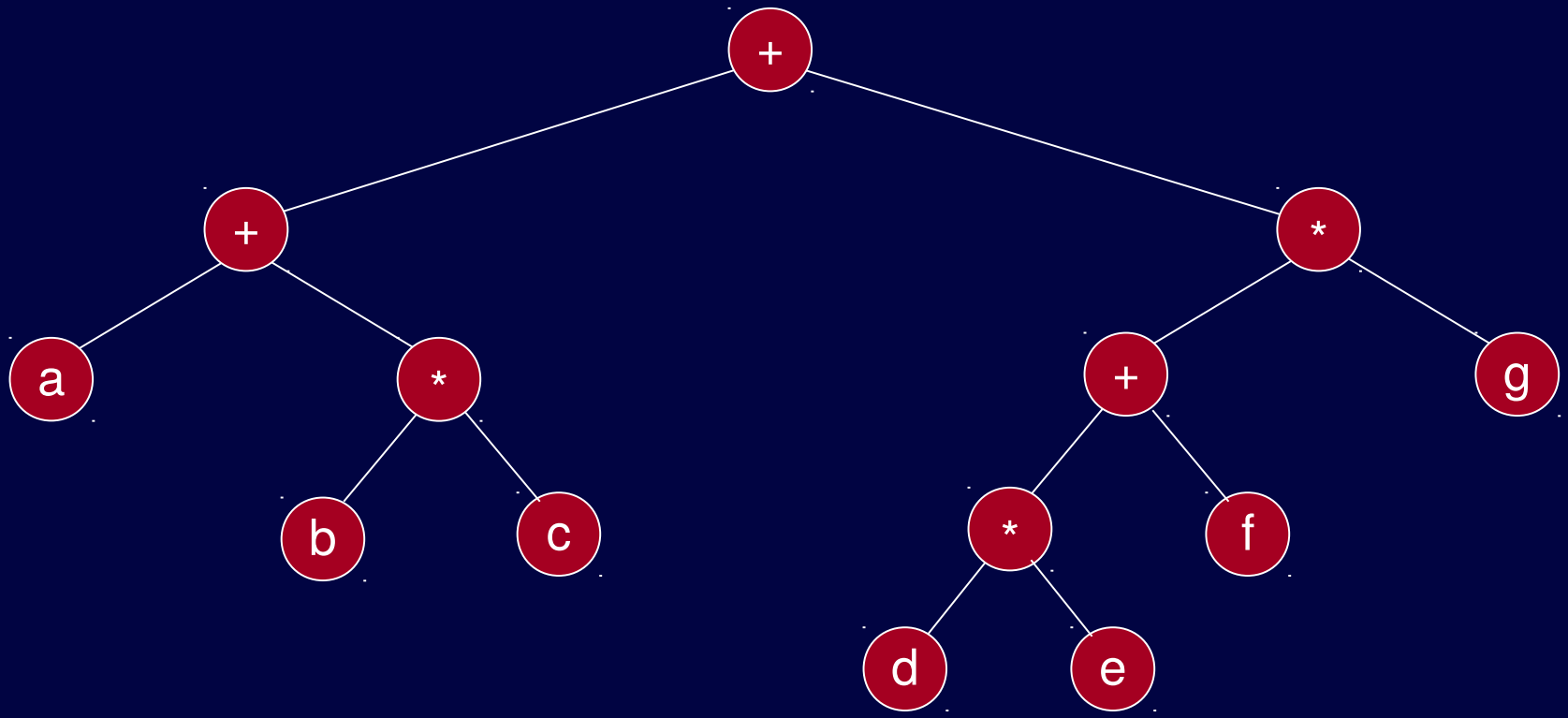
Expression Tree

- This is not necessary. A unary operator (!, e.g.) will have only one subtree.



Expression Tree

- Inorder traversal yields: a+b*c+d*e+f*g

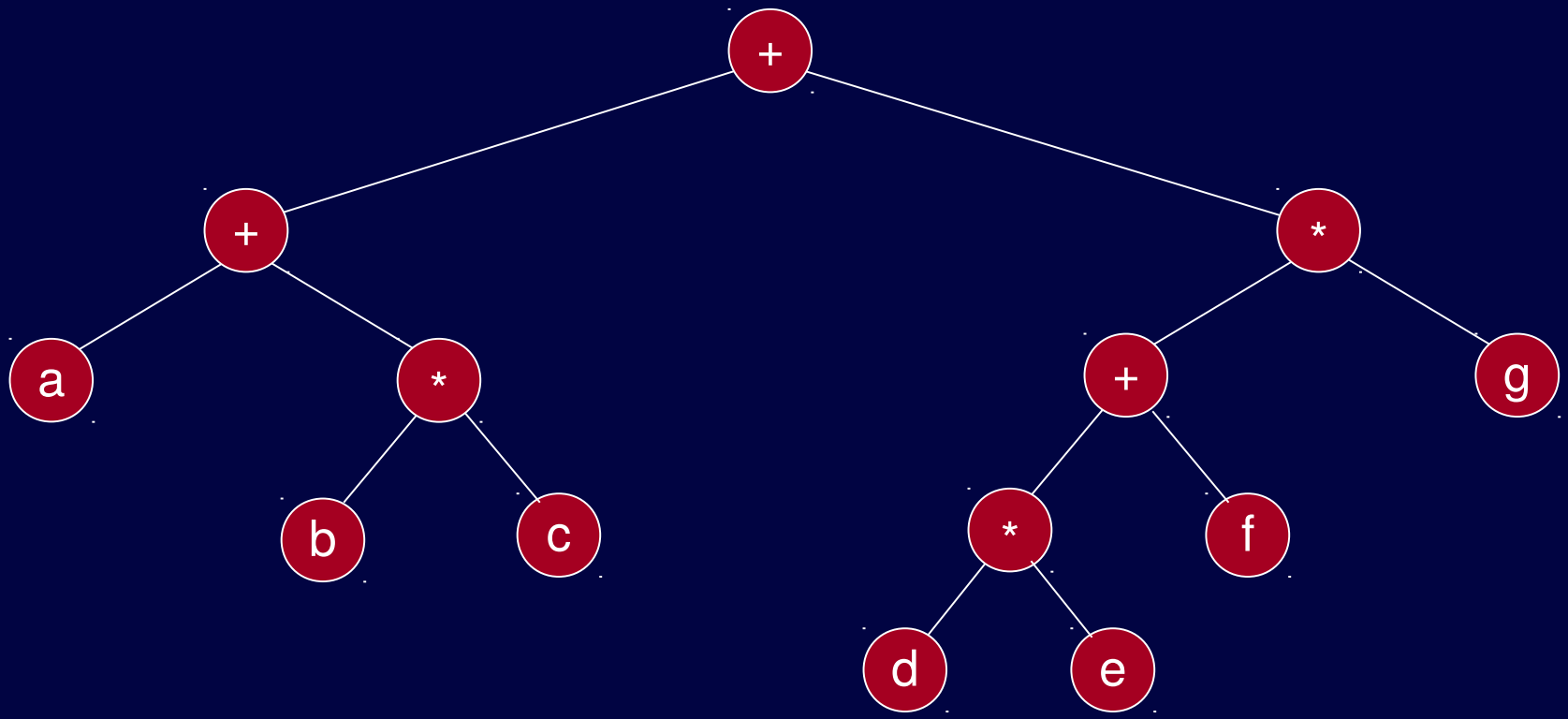


Enforcing Parenthesis

```
/* inorder traversal routine using the parenthesis
   */
void inorder(TreeNode<int>* treeNode)
{
  if( treeNode != NULL )
  {
    if(treeNode->getLeft() != NULL && treeNode-
  >getRight() != NULL) //if not leaf
      cout<<"(";
    inorder(treeNode->getLeft());
    cout << *(treeNode->getInfo())<<" ";
    inorder(treeNode->getRight());
    if(treeNode->getLeft() != NULL && treeNode-
  >getRight() != NULL) //if not leaf
      cout<<")";
  }
}
```

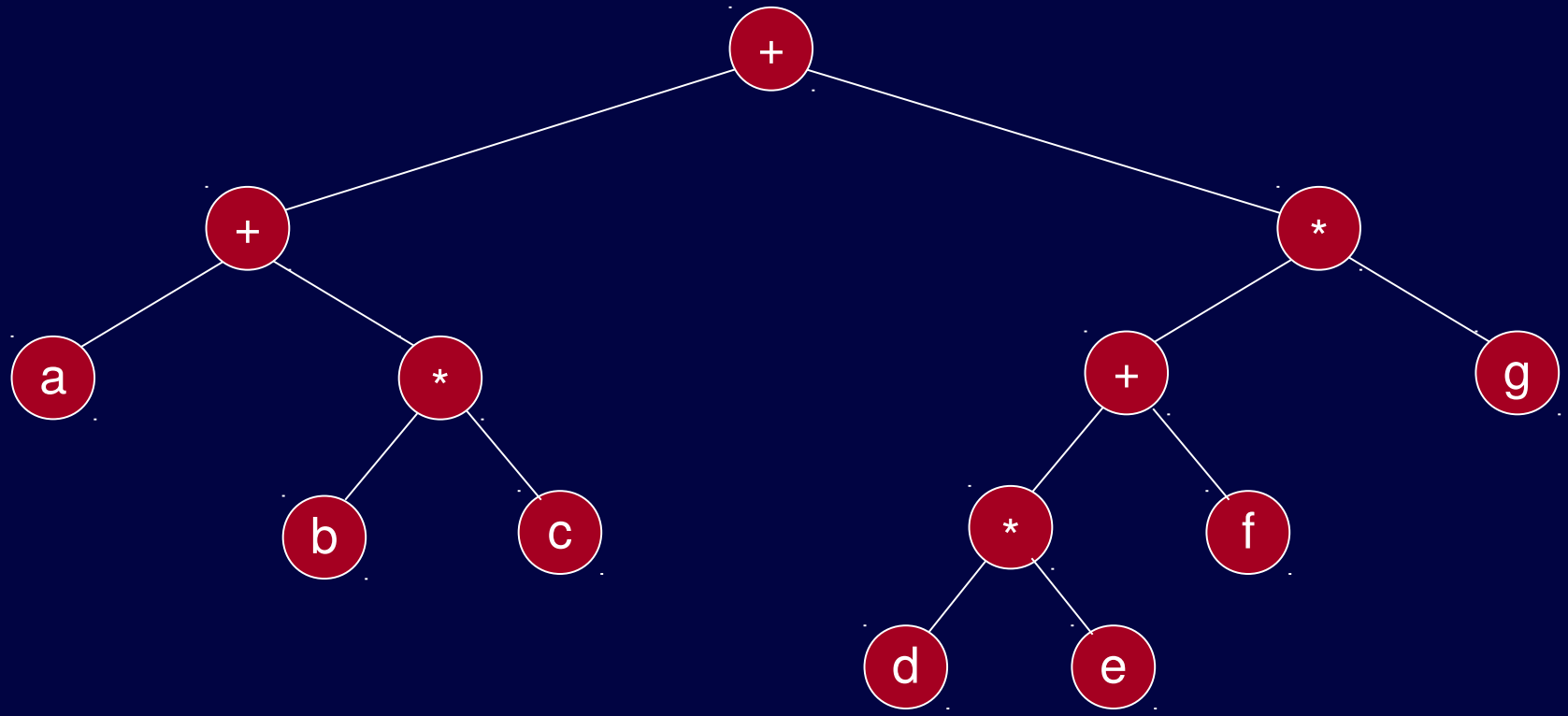
Expression Tree

- Inorder : (a+(b*c))+(((d*e)+f)*g)



Expression Tree

- Postorder traversal: a b c * + d e * f + g * +
  which is the postfix form.



Constructing Expression Tree

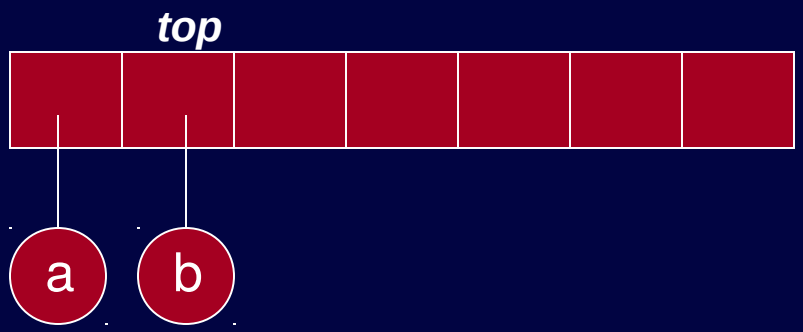
- Algorithm to convert postfix expression into an expression tree.
- We already have an expression to convert an infix expression to postfix.
- Read a symbol from the postfix expression.
- If symbol is an operand, put it in a one node tree and push it on a stack.
- If symbol is an operator, pop two trees from the stack, form a new tree with operator as the root and T_1 and T_2 as left and right subtrees and push this tree on the stack.

Constructing Expression Tree

- a b + c d e + * *

Constructing Expression Tree

- a b + c d e + * *

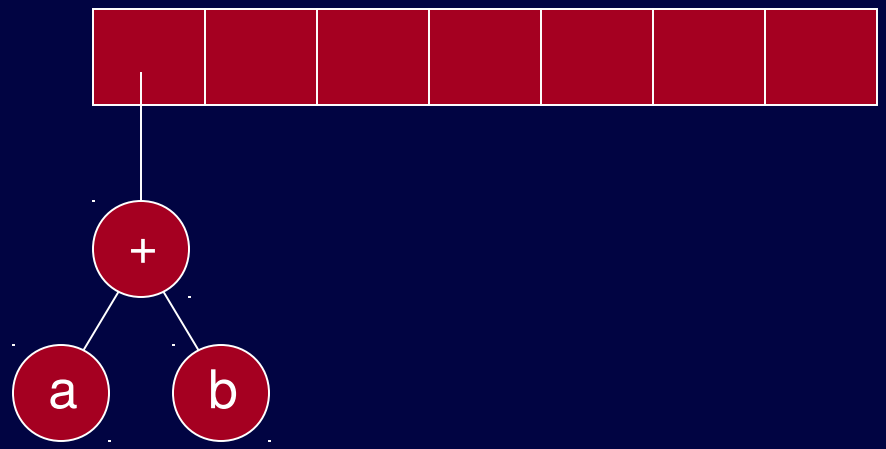


If symbol is an operand, put it in a one node tree and push it on a stack.

Stack is growing left to right

Constructing Expression Tree

- a b + c d e + * *

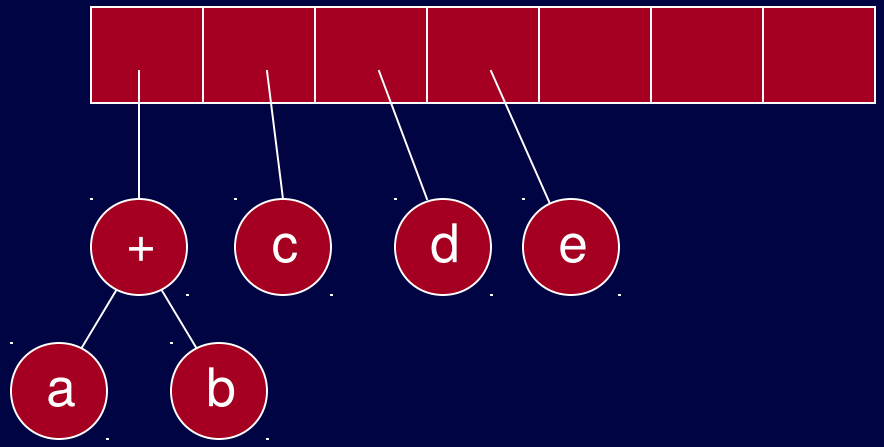


If symbol is an operator, pop two trees from the stack, form a new tree with operator as the root and T_1 and T_2 as left and right subtrees and push this tree on the stack.

Stack is growing left to right

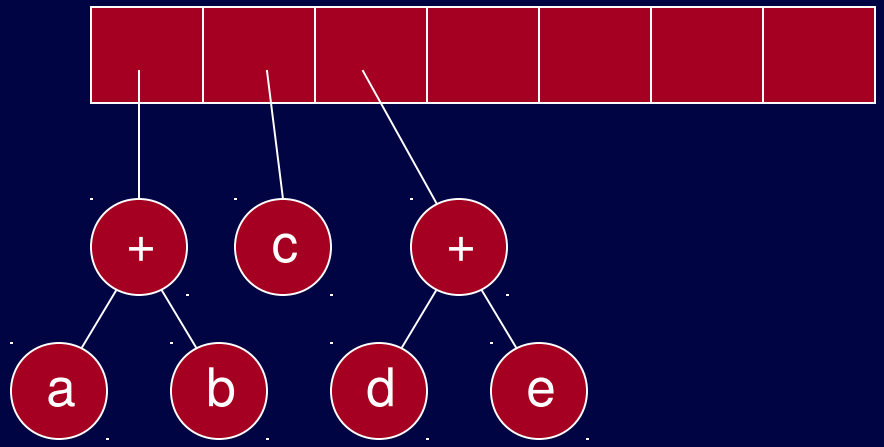
Constructing Expression Tree

- a b + c d e + * *



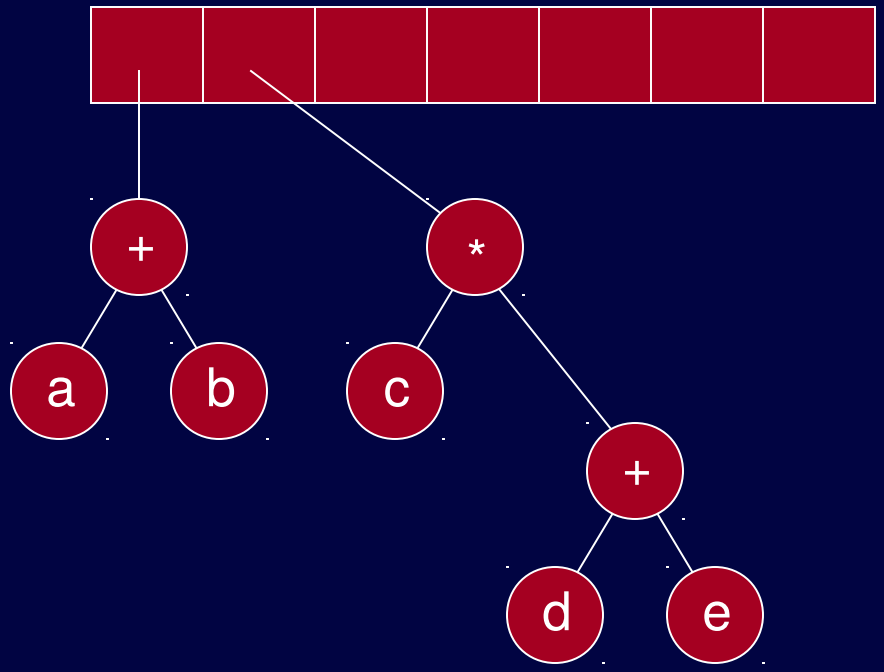
Constructing Expression Tree

- a b + c d e + * *



Constructing Expression Tree

- a b + c d e + * *



Constructing Expression Tree

- a b + c d e + * *