ECE250 Lab 3
Expression Trees

Tiuley Alguindigue

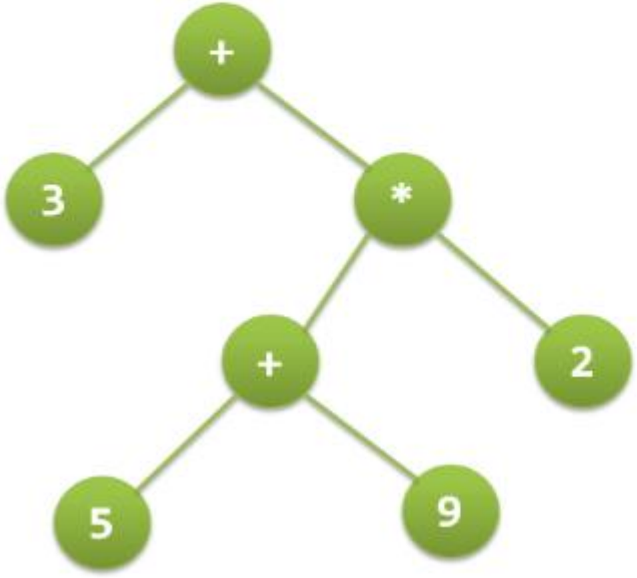
March 5th, 2012

Overview

- Expression Trees
- Project 3 Classes
- Recursive approach
- Reverse polish and in-fix notation
- Expression Class
- Expression Tree Class
- Where to start?
- Adding parenthesis to In-fix function
- Testing

Expression Trees

Expression Trees are special kind of binary trees:



Project 3

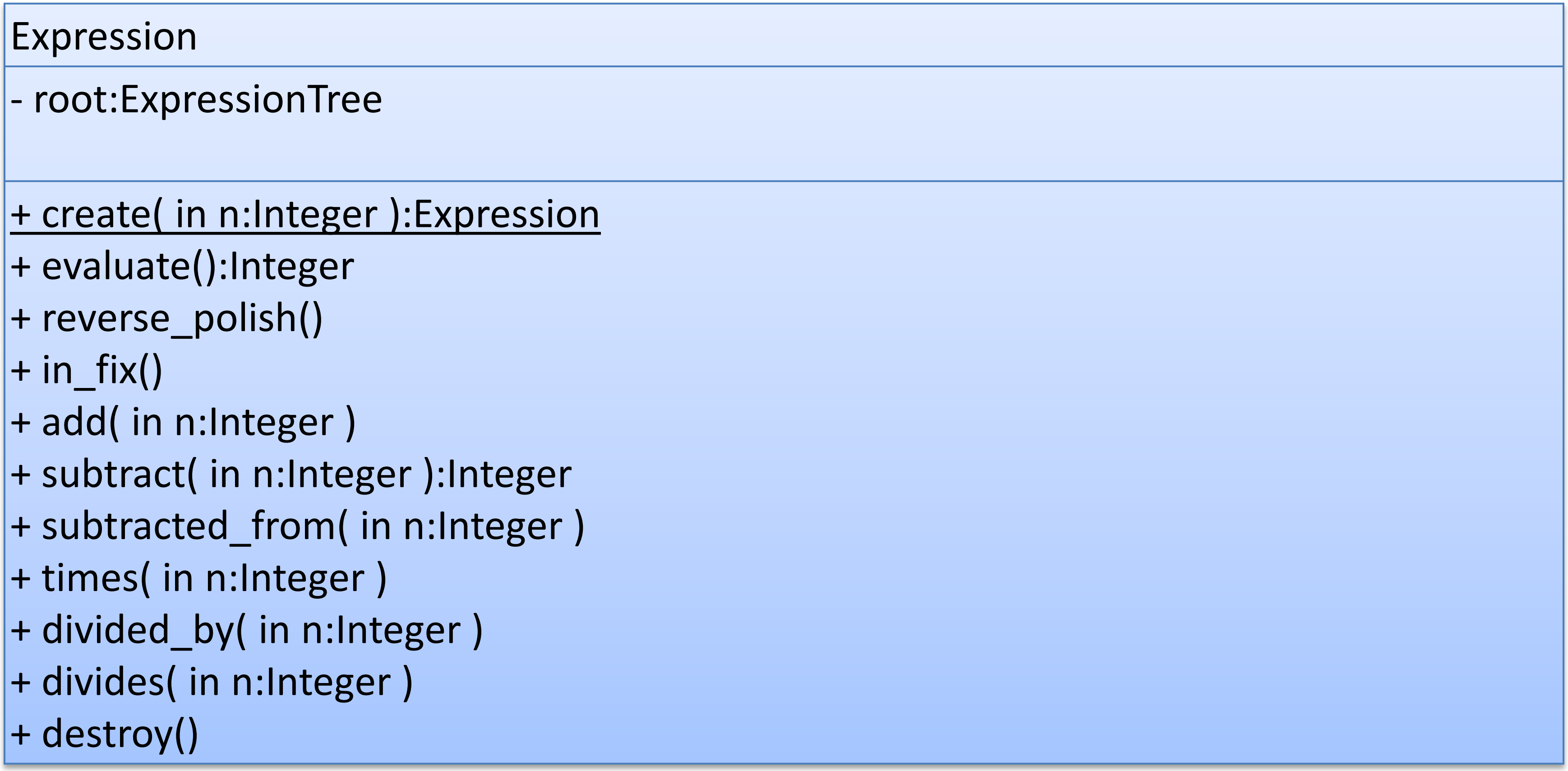
- In project 3, you will implemented an expression tree with classes:
  - Expression.h
  - ExpressionTree.h
- You will use recursion to evaluate expressions, and to print the expression in polish notation and in-fix notation.

Reverse Polish and In-fix Notation

The following give reverse-Polish and in-fix forms for the same expression:

Reverse Polish	**In-fix**
3	3
3 5 +	3 + 5
3 5 + 2 -	3 + 5 - 2
3 5 - 2 +	3 - 5 + 2
3 5 2 + -	3 - (5 + 2)
3 5 2 * +	3 + 5 * 2
3 5 2 + *	3 * (5 + 2)

Expression Class

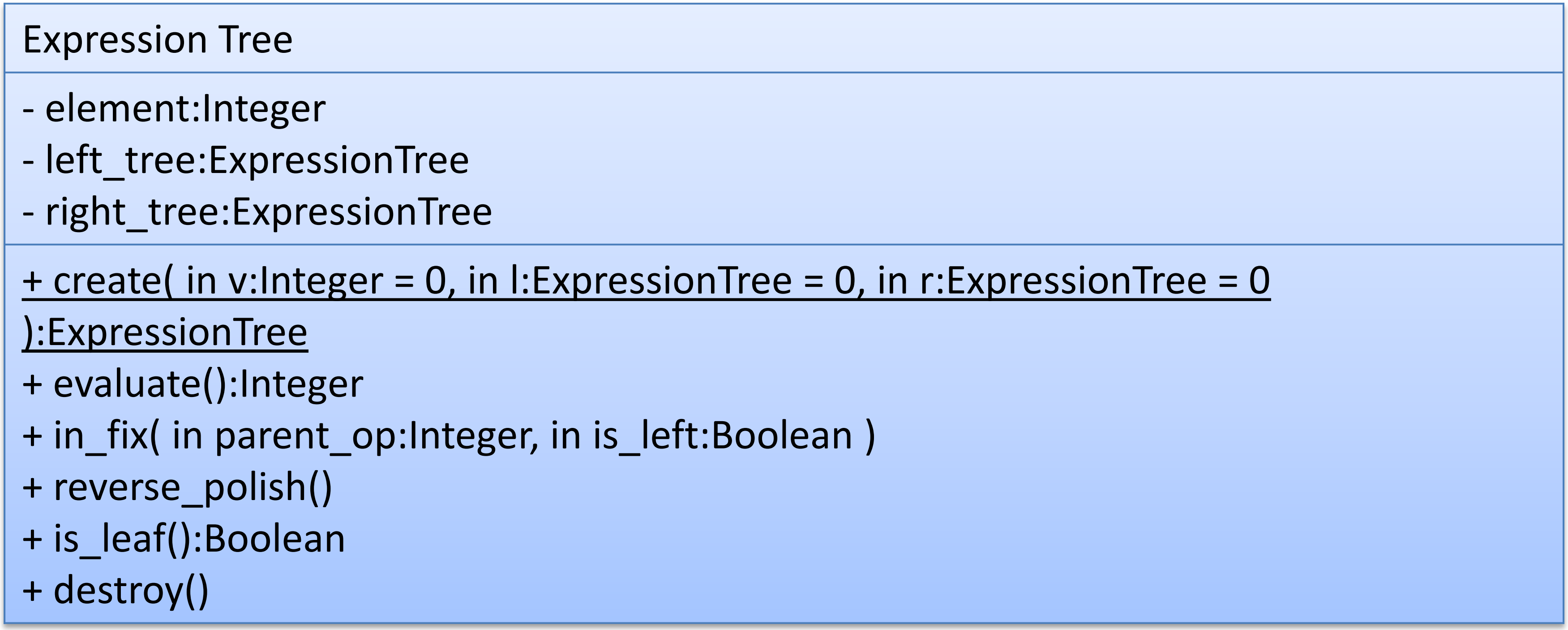


Expression

- root:ExpressionTree

+ create(in n:Integer):Expression
+ evaluate():Integer
+ reverse_polish()
+ in_fix()
+ add(in n:Integer)
+ subtract(in n:Integer):Integer
+ subtracted_from(in n:Integer)
+ times(in n:Integer)
+ divided_by(in n:Integer)
+ divides(in n:Integer)
+ destroy()

Expression Tree Class

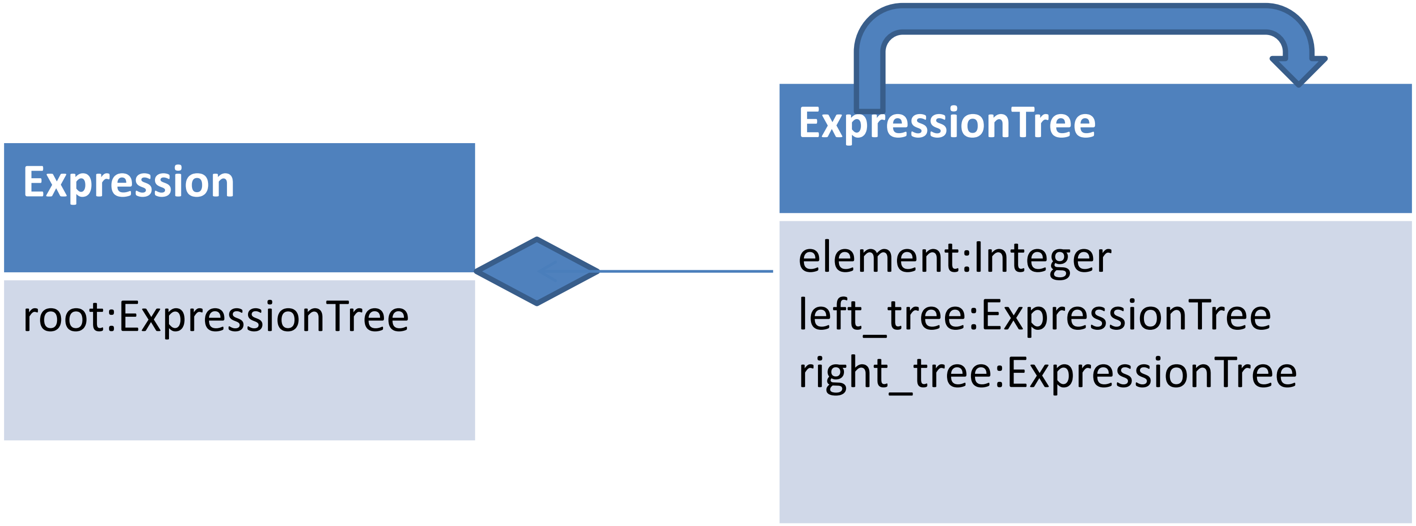


Expression Tree

- element:Integer
- left_tree:ExpressionTree
- right_tree:ExpressionTree

+ create(in v:Integer = 0, in l:ExpressionTree = 0, in r:ExpressionTree = 0):ExpressionTree
+ evaluate():Integer
+ in_fix(in parent_op:Integer, in is_left:Boolean)
+ reverse_polish()
+ is_leaf():Boolean
+ destroy()

Expression and Expression Tree



Expression

root:ExpressionTree

ExpressionTree

element:Integer
left_tree:ExpressionTree
right_tree:ExpressionTree

Expression Tree Class

Constructor and destructor:

ExpressionTree(in v:Integer = 0, in l:ExpressionTree = 0, in r:ExpressionTree = 0):ExpressionTree

~ExpressionTree()

Accessors

evaluate():Integer - evaluate expression below the current node

in_fix(in parent_op:Integer, in is_left:Boolean) – prints the in-fix notation for expression below the current node

reverse_polish()) – prints the reverse polish notation for expression below the current node

is_leaf():Boolean – determines if node is a leaf node

Expression Class

Constructor and Destructor:

Expression(in n:Integer):Expression

~Expression()

Functions below - call same function on root Expression Tree:

reverse_polish()
evaluate():Integer
in_fix()

Four types of insert:

add(in n:Integer) - adds new Expression Tree, with left child equal to existing tree and right child containing n. The value in the node is operator “+”

subtract(in n:Integer):Integer
subtracted_from(in n:Integer)
times(in n:Integer)
divided_by(in n:Integer)
divides(in n:Integer)

Where to start?

- Expression Tree – constructor, destructor
- Expression – mutators to insert nodes (add, subtract , etc.)
- Expression Tree – accessors

  is_leaf()

  evaluate()

  reverse_polish()

  in_fix(int , bool) – tip: first ignore parenthesis rules.
- Expression – accessors (call same methods on root Expression Tree)

  evaluate()

  reverse_polish()

  in_fix()

Expression Tree
Adding Parenthesis to In-fix function

Printing Expressions in In-Fix Form

The requirements for printing an expression in in-fix form are:

1. All operators must be printed with one space on either side.
2. If an addition or subtraction operation is a child of a multiplication or division operation(*) , it must be surrounded by parentheses. There is no space between parentheses and the operands which they surround.
3. If an addition or subtraction is the right sub-operation of a subtraction, it must be surrounded by parentheses(*).
4. If a multiplication or division is the right sub-operation of a division, it must be surrounded by parentheses.

(*) In-fix will need to know the value of its parent, and the type of branch (left or light)

Testing Files

File e01.in - will give you 30 % of the marks.

Must match output in e01.out

```
new
add 3
add 7
subtract 5
times 4
divided_by 2
evaluate 10
reverse_polish
delete
```

….. and more