Jacob Ciurca

Dr. Lu

Lab 4

Cosc 320

9/30/2020

**Prelab:** reviewed binary tree traversal as well as reviewed given code

**Lab:**

**Question:** What relationship exists among these scans and prefix and postfix notation for the expression?

**Answer:** they both output the binary operators either before or after the operand children. For example both postfix and post-order expressions will be the same for a given binary tree.

**expTree.h:**

#ifndef EXPTREE\_H\_INCLUDED

#define EXPTREE\_H\_INCLUDED

#include <iostream>

#include <sstream>

#include <iomanip>

#include <string>

#include <queue>

#include <stack>

#include <cstring>

#ifndef NULL

#include <cstddef>

#endif // NULL

#include "d\_tnode.h"

tnode<char> \*buildExpTree1()

{

tnode<char> \*root, \*b, \*c;

b = new tnode<char> ('a');

c = new tnode<char> ('b');

root = new tnode<char> ('\*', b, c);

return root;

}

tnode<char> \*buildExpTree2()

{

tnode<char> \*root, \*b, \*c, \*d, \*e;

d = new tnode<char> ('a');

e = new tnode<char> ('b');

c = new tnode<char> ('c');

b = new tnode<char> ('+', d, e);

root = new tnode<char> ('\*', b, c);

return root;

}

tnode<char> \*buildExpTree3(){

tnode<char> \*root, \*b, \*c, \*d, \*e, \*f, \*g, \*h, \*i;

h = new tnode<char> ('a');

i = new tnode<char> ('b');

g = new tnode<char> ('c');

e = new tnode<char> ('d');

c = new tnode<char> ('e');

f = new tnode<char> ('+', h, i);

d = new tnode<char> ('\*', f, g);

b = new tnode<char> ('/', d, e);

root = new tnode<char> ('-', b, c);

return root;

}

template <typename T>

void PreorderOutput(tnode<T> \*t, const string& separator = " ")

{

// the recursive scan terminates on a empty subtree

if (t != NULL)

{

cout << t->nodeValue << separator;

PreorderOutput(t->left, separator); // descend left

PreorderOutput(t->right, separator); // descend right

// output the node

}

}

void prefixOutput(tnode<char> \*exp){

if(exp != NULL){

cout << exp ->nodeValue << " ";

prefixOutput(exp->left);

prefixOutput(exp->right);

}

}

tnode<char> \*buildExpTree(const string& exp)

{

// newnode is the address of the root of subtrees we build

tnode<char> \*newnode, \*lptr, \*rptr;

char token;

// subtrees go into and off the stack

stack<tnode<char> \*> s;

int i = 0;

// loop until i reaches the end of the string

while(i != exp.length())

{

// skip blanks and tabs in the expression

while (exp[i] == ' ' || exp[i] == '\t')

i++;

// if the expression has trailing whitespace, we could

// be at the end of the string

if (i == exp.length())

break;

// extract the current token and increment i

token = exp[i];

i++;

// see if the token is an operator or an operand

if (token == '+' || token == '-' || token == '\*' || token == '/')

{

// current token is an operator. pop two subtrees off

// the stack.

rptr = s.top();

s.pop();

lptr = s.top();

s.pop();

// create a new subtree with token as root and subtees

// lptr and rptr and push it onto the stack

newnode = new tnode<char>(token,lptr,rptr);

s.push(newnode);

}

else // must be an operand

{

// create a leaf node and push it onto the stack

newnode = new tnode<char>(token);

s.push(newnode);

}

}

// if the expression was not empty, the root of the expression

// tree is on the top of the stack

if (!s.empty())

return s.top();

else

return NULL;

}

#endif // EXPTREE\_H\_INCLUDED

**Lab04.cpp:**

#include <iostream>

#include <string>

#include <stack>

#include "d\_tnode.h"

#include "d\_tnode1.h"

#include "expTree.h"

#include "infixtopostfix.h"

#include "d\_tnode.h"

#include "d\_expsym.h"

#include "d\_except.h"

using namespace std;

int main()

{

tnode<char> \*t1 = buildExpTree1();

tnode<char> \*t2 = buildExpTree2();

tnode<char> \*t3 = buildExpTree3();

cout << "Pre-order and post-order output for the first binary expression tree:\n";

PreorderOutput(t1);

cout << endl;

postorderOutput(t1);

cout << endl;

cout << "Pre-order and post-order output for the second binary expression tree:\n";

PreorderOutput(t2);

cout << endl;

postorderOutput(t2);

cout << endl;

cout << "Pre-order and post-order output for the third binary expression tree:\n";

PreorderOutput(t3);

cout << endl;

postorderOutput(t3);

cout << endl;

cout << endl;

cout << "Prefix of tree 2: ";

prefixOutput(t2);

cout << endl;

cout << "Please enter an expression to put in a tree.\n";

string tree;

cin >> tree;

infix2Postfix in;

in.setInfixExp(tree);

string post\_tree;

post\_tree = in.postfix();

tnode<char> \*t4 = buildExpTree(post\_tree);

cout << "Prefix and Postfix output of the tree.\n";

prefixOutput(t4);

cout << endl;

postorderOutput(t4);

cout << endl;

cout << "Tree Display.\n";

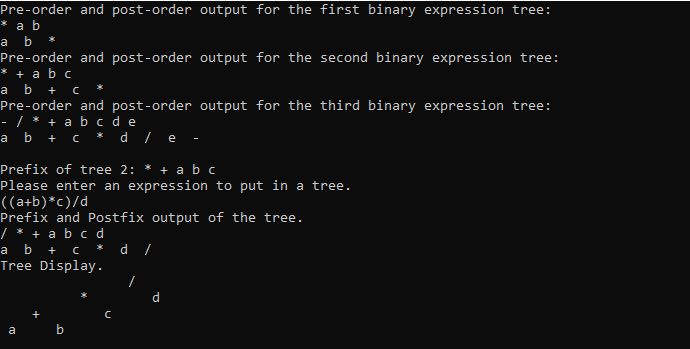
displayTree(t4, 2);

cout << endl;

return 0;

}

**Sample Output:**



**Postlab:** This lab was very good at teaching us more on binary trees and binary expression trees and how to build and traverse them. This lab took me about 1 hour 20 minutes to complete. I completed this lab by myself with no help.