Michael Smith

Mondays (Section 2) – Lab 1

CSE 330 Data Structures

Winter 2017

Infix-to-Postfix Conversions

1. Status
   1. The lab to convert infix-to-postfix expressions is 100% complete and fully operational.
2. Complexity Analysis
   1. Main has a big-Oh notation of (n^4+n+9c) so therefore main has a O(n^4). The storage complexity of main() is
   2. Prec has a big-Oh notation of (6c) so therefore prec() has a O(1). The storage complexity of the prec() is
3. Source Code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Michael Smith

\* expr.cpp

\* 01/09/2017

\* This program converts infix expression to a postfix expression.

\* This file contains two functions. The main function which holds the

\* algorithm for the converting of the infix expressions to the

\* post fix expression. The second function is a precedence checker

\* which checks operators and gives them a precedence level to evaluate

\* their order.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stack>

#include <iostream>

int prec(char,int);

using namespace std;

stack<char> operators;

// Main file. Holds algorithm to convert infix to postfix expressions while

// using function prec to check operator precedence.

int main () {

char ch;

cin.get(ch);

while ( !cin.eof() ) {

while ( ch != '\n' ) { //Loop Through Input Line

if ( ch == ' ') { //Skip white Spaces

cin.get(ch);

continue; }

if ((ch >= '0' && ch <= '9') || (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {

cout << ch; }

else if (ch == ')') {

while (!operators.empty() && operators.top() != '(') {

cout << operators.top();

operators.pop(); }

if (!operators.empty())

operators.pop();

else

cout << "There was an error no matching '(' was found" << endl; }

else if (ch == '+' || ch =='-' || ch == '\*' || ch == '/' || ch == '(') {

if (operators.empty() || prec(operators.top(),0) < prec(ch,1))

operators.push(ch);

else {

while (!operators.empty() && (prec(operators.top(),0) >= prec(ch,1))) {

cout << operators.top();

operators.pop(); }

operators.push(ch); } }

else {

if (ch != '\n' ) {

cout << "There was an error with the input" << endl;

} }

cin.get(ch);}

while (!operators.empty()) {

cout << operators.top();

operators.pop(); }

cout << endl;

cin.get(ch);

}

return 0; }

// Checks the precedence of the operators to determine order of operations and whether were testing

// the stack or a passed input char.

int prec(char input,int flag) {

int output;

if (input == '\*' || input == '/') //Sets \* and / as the highest precedence at 2

output = 2;

else if (input == '+' || input == '-') //Sets + and - as the middle precedence at 1

output = 1;

else if ( input == '(' ) { //Sets the ( as the lowest precedence when inside the stack and sets it to the highest outside of the stack

if ( flag == 1 )

output = 3;

else

output = 0; }

else

cout << "There was an invalid operator" << endl;

return output;

}

1. Sample Run
2. Script started on Sun 22 Jan 2017 10:51:36 PM PST
3. ]0;mike@mike-MacBookPro ~/Desktop/projects/Lab1[01;32mmike@mike-MacBookPro[00m [01;34m~/Desktop/projects/Lab1 $[00m ls
4. [0m[01;32ma.out[0m [01;32mexpr[0m expr.o Lab1.cbp Lab1.layout typescript
5. [01;34mbin[0m expr.cpp exproutput.doc Lab1.depend [01;34mobj[0m
6. ]0;mike@mike-MacBookPro ~/Desktop/projects/Lab1[01;32mmike@mike-MacBookPro[00m [01;34m~/Desktop/projects/Lab1 $[00m g++ expr.cpp
7. ]0;mike@mike-MacBookPro ~/Desktop/projects/Lab1[01;32mmike@mike-MacBookPro[00m [01;34m~/Desktop/projects/Lab1 $[00m ./a.out
8. a+b\*c
9. abc\*+
10. (a-2)/(5-d)
11. a2-5d-/
12. a+((b-c)\*d)/e
13. abc-d\*e/+
14. ]0;mike@mike-MacBookPro ~/Desktop/projects/Lab1[01;32mmike@mike-MacBookPro[00m [01;34m~/Desktop/projects/Lab1 $[00m exit
15. Script done on Sun 22 Jan 2017 10:52:25 PM PST