CSCI 301 Computer Science II Summer 2023

**Assignment 5 – Infix Expression Calculators**

Due Date: 11:50 pm July 14, Friday

# Introduction

This project consists of designing and implementing a program of an infix calculator by using the C++ programming language. This program is a calculator that basically takes an infix mathematical expression as an input and converts it to a postfix expression and also returns accurately the final result of the expression.It supports the operators +, -, \*, and /. It checks for the well-formedness and balanced brackets of the expression and uses the infix-to-postfix conversion algorithm to evaluate the expression. The program prompts the user to enter infix expressions and evaluates them until the user decides to exit. The program is implemented using a stack algorithm in the class design.

# Data Structure

# The program uses several data structures. Among them, we have implemented a few strings for the infixExpression, and also for the converterToPostfix. We have the integer data type implemented for the function getPrecedence to get the precedent term. We also have a couple of booleans in the class to check if the term is an operator, if expression is well formed,and also if it is balanced.

# Functions

We have design a few functions to successfully implement the program among them we have:

* A function to check if the input is an operator and then returns it
* A function to get the precedent term of the inputted expression
* A function to check if the expression entered is well formed
* A function to verify of the expression is balanced
* A function that converts infix expression to postfix expression
* A function that evaluates the converted postfix expression

# Structure chart of the main program



# Code list

/\*\*

\* problem description: Design and implement a class of infix calculator.

\* Name: Algassimou Diallo

\* startID: el8524jv

\* Instructor: Jie Meichsner

\* Due date: 07/14/2023

\*/

#include <iostream>

#include <string>

#include <stdexcept>

#include <stack>

using namespace std;

/\*\*

\* InfixCalculator class represents an infix expression calculator.

\* The calculator supports basic arithmetic operators (+, -, \*, /),

\* single-digit operands, and parentheses.

\*/

class InfixCalculator {

private:

string infixExpression; // Data member to store the infix expression

bool isOperator(char ch) {

return (ch == '+' || ch == '-' || ch == '\*' || ch == '/');

}

int getPrecedence(char op) {

if (op == '+' || op == '-')

return 1;

else if (op == '\*' || op == '/')

return 2;

else

return 0;

}

/\*\*

\* Checks if an infix expression is well formed.

\* @return True if the expression is well formed, False otherwise.

\*/

bool isWellFormed() {

int openCount = 0;

int closeCount = 0;

for (char ch : infixExpression) {

if (ch == '(')

openCount++;

else if (ch == ')')

closeCount++;

}

return (openCount == closeCount);

}

/\*\*

\* Checks for balanced brackets in the infix expression.

\* @return True if brackets are balanced, False otherwise.

\*/

bool isBalanced() {

stack<char> brackets;

for (char ch : infixExpression) {

if (ch == '(')

brackets.push(ch);

else if (ch == ')') {

if (brackets.empty())

return false;

brackets.pop();

}

}

return brackets.empty();

}

/\*\*

\* Converts an infix expression to postfix notation.

\* @return The postfix expression as a string.

\*/

string convertToPostfix() {

stack<char> operators;

string postfixExpression;

for (char ch : infixExpression) {

if (isalnum(ch)) {

postfixExpression += ch;

} else if (isOperator(ch)) {

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

postfixExpression += operators.top();

operators.pop();

}

operators.push(ch);

} else if (ch == '(') {

operators.push(ch);

} else if (ch == ')') {

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

postfixExpression += operators.top();

operators.pop();

}

operators.pop(); // Discard '('

}

}

while (!operators.empty()) {

postfixExpression += operators.top();

operators.pop();

}

return postfixExpression;

}

/\*\*

\* Evaluates a postfix expression and returns the result.

\* @param postfixExpression The postfix expression to evaluate.

\* @return The result of the evaluation.

\*/

int evaluatePostfix(const string& postfixExpression) {

stack<int> operands;

for (char ch : postfixExpression) {

if (isalnum(ch)) {

operands.push(ch - '0');

} else if (isOperator(ch)) {

int operand2 = operands.top();

operands.pop();

int operand1 = operands.top();

operands.pop();

int result;

switch (ch) {

case '+':

result = operand1 + operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '/':

if (operand2 == 0)

throw runtime\_error("Division by zero error");

result = operand1 / operand2;

break;

}

operands.push(result);

}

}

return operands.top();

}

public:

/\*\*

\* Default constructor for InfixCalculator class.

\*/

InfixCalculator() {

infixExpression = "";

}

/\*\*

\* Sets the data member with an infix expression after checking if it is valid.

\* @param expression The infix expression to set.

\* @return True if the expression is valid and set successfully, False otherwise.

\*/

bool setExpression(const string& expression) {

if (!isWellFormed() || !isBalanced())

return false;

infixExpression = expression;

return true;

}

/\*\*

\* Evaluates the expression stored in the data member.

\* @return The result of the evaluation.

\*/

int evaluateExpression() {

string postfixExpression = convertToPostfix();

cout << "The postfix expression is: " << postfixExpression << endl;

return evaluatePostfix(postfixExpression);

}

};

int main() {

// Create an instance of InfixCalculator

InfixCalculator calculator;

// Prompt the user to enter an expression

while (true) {

string expression;

cout << "Enter an infix expression (q to quit): ";

getline(cin, expression);

if (expression == "q")

break;

if (!calculator.setExpression(expression)) {

cout << "Invalid expression. Please enter a well-formed and balanced infix expression." << endl;

continue;

}

try {

int result = calculator.evaluateExpression();

cout << "Result: " << result << endl;

} catch (const exception& e) {

cout << "Error: " << e.what() << endl;

}

}

return 0;

}

User Document

To successfully run this program, the user must enter only single digit numbers and use one more of the following operators: +, -, \*, and /. The program will request an infix expression from the user which will be used as input. The program will then convert it to a postfix expression, evaluate the final result and print the output.The user can use parenthesis within the infix expression but it should make sure that all opened parenthesis have closed parenthesis equivalently. No other characters or symbols are allowed in this program. Program will keep running until the user decides to quit by entering the letter “q”. This program is located in the centOS directory: el8524jv/csci301/project5

To successfully compile this program, use the command g++ -std=c++11 -g infixCalc.cpp -o main. Then you run ./main.

# Test cases

| Tests | Inputs | outputs |
| --- | --- | --- |
| 1. Valid expressions | 2\*(2+4)/2 | The postfix expression is:224+\*2/  Result: 6 |
| 1. Imbalance expression- missing closing parenthesis | 3+4(4-2 | Invalid expression. Please enter a well-formed and balanced infix expression. |
| 1. Imbalance expression- missing opening parenthesis | 3-2) | Invalid expression. Please enter a well-formed and balanced infix expression. |
| 1. Invalid expression-two digit term entered | 45+4  3+45 | The postfix expression is: 454+  Result: 9  The program ignores the first digit(4)  The postfix expression is: 345+  Result: 9  The program ignores the last digit |
| 1. No operator in the expression | 3 4  53 | The postfix expression is: 34  Result: 4  The postfix expression is: 53  Result: 3 |

# Test scripts

The following scripts represent the tests described in the table above. They was run in the centOS server and you can also find them in this directory: el8524jv/csci301/project5.

* **Test Script1**

Script started on Fri 14 Jul 2023 08:34:22 PM CDT

^[]0;el8524jv@csci4:~/csci301/project5^G^[[?1034h[el8524jv@csci4 project5]$ g++ /home/STCLOUDSTATE/el8524jv/csci301/

Enter an infix expression (q to quit): 2\*(2+4)/2^M

The postfix expression is: 224+\*2/^M

Result: 6^M

Enter an infix expression (q to quit): 3-4^M

The postfix expression is: 34-^M

Result: -1^M

Enter an infix expression (q to quit): 5+4?^H ^H/3^M

The postfix expression is: 543/+^M

Result: 6^M

Enter an infix expression (q to quit): q^M

^[]0;el8524jv@csci4:~/csci301/project5^G[el8524jv@csci4 project5]$ exit^M

Script done on Fri 14 Jul 2023 08:36:04 PM CDT

* **Test** **Script 2**

Script started on Fri 14 Jul 2023 08:40:00 PM CDT

^[]0;el8524jv@csci4:~/csci301/project5^G^[[?1034h[el8524jv@csci4 project5]$ ./main^M

Enter an infix expression (q to quit): 3+4(4-2^M

Invalid expression. Please enter a well-formed and balanced infix expression.^M

Enter an infix expression (q to quit): 4+49^H ^H(4^M

Invalid expression. Please enter a well-formed and balanced infix expression.^M

Enter an infix expression (q to quit): q^M

^[]0;el8524jv@csci4:~/csci301/project5^G[el8524jv@csci4 project5]$ exit^M

Script done on Fri 14 Jul 2023 08:41:26 PM CDT

* **Test Script3**

Script started on Fri 14 Jul 2023 08:44:28 PM CDT

[el8524jv@csci4 project5]$ ./main

Enter an infix expression (q to quit): 3-2)

Segmentation fault (core dumped)

[el8524jv@csci4 project5]$ 3+5

bash: 3+5: command not found...

[el8524jv@csci4 project5]$ 3+5./main

Enter an infix expression (q to quit): 5+)-3

Segmentation fault (core dumped)

[el8524jv@csci4 project5]$ exit

Script done on Fri 14 Jul 2023 08:46:05 PM CDT

* **Test Script 4**

Script started on Fri 14 Jul 2023 08:48:36 PM CDT

[el8524jv@csci4 project5]$ ./main

Enter an infix expression (q to quit): 45+4

The postfix expression is: 454+

Result: 9

Enter an infix expression (q to quit): 3+45

The postfix expression is: 345+

Result: 9

Enter an infix expression (q to quit): 345-2

The postfix expression is: 3452-

Result: 3

Enter an infix expression (q to quit): q

[el8524jv@csci4 project5]$ exit

Script done on Fri 14 Jul 2023 08:49:39 PM CDT

* **Test Script5**

Script started on Fri 14 Jul 2023 08:50:41 PM CDT

[el8524jv@csci4 project5]$ ./main

Enter an infix expression (q to quit): 3 4

The postfix expression is: 34

Result: 4

Enter an infix expression (q to quit): 53

The postfix expression is: 53

Result: 3

Enter an infix expression (q to quit): 6 7

The postfix expression is: 67

Result: 7

Enter an infix expression (q to quit): q

[el8524jv@csci4 project5]$ exit

Script done on Fri 14 Jul 2023 08:51:48 PM CDT

# Summary

In this project, we have implemented a calculator that takes an infix mathematical expression, converts it to a postfix expression and returns it. It also calculates and gives the final result of these expressions as an output to the user. The program prompts the user to enter an expression and will keep doing it until the user decides to quit by entering the letter “q”. However, there are some restrictions for this program to work as intended among those: the user must use a single digit term only. For example if the user enters 34, the program will only consider the last digit of the entered term which is 4 in this case. User must make sure to enter at least on of the following operators: +,-,\*,/. Parenthesis can be also used in this program but the user is required to close any opened parenthesis otherwise, the expression won’t be balanced and will fail consequently. Finally, the program is running perfectly in centOS with no failures registered.