Quinn Roemer

CISP - 430

Assignment 5.42

3/1/2018

Program - Infix to Postfix/Evaluator Implementation

Description:

The goal for this assignment was to implement the functions that we used for our hand executions in the previous assignment. We had the option of using some code written by the professor or writing our own. I wrote my own. My code uses the Standard Template Library for both the stack and queue containers.

Source Code:

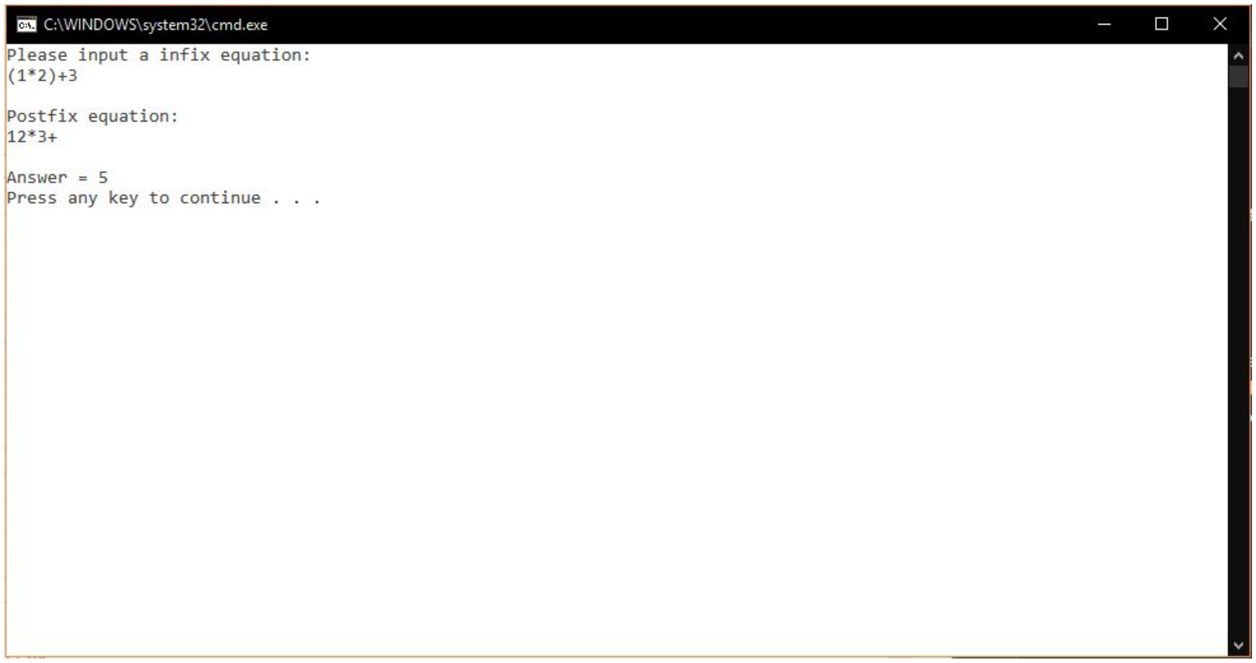
```
//Code written by Quinn Roemer, based on code by Professor Ross.
#include <iostream>
#include <stack>
#include <queue>
#include <string>

using namespace std;

//Token Class
class token {
public:
    char data;
    int type;
    int preference;

    void setData(char);

    void outputData()
    {
        cout << data;
    }
};

void token::setData(char temp)
{
    switch (temp)
    {
    case '*':
        data = temp;
        type = 0;
        preference = 3;
        break;
    case '/':
        data = temp;
        type = 0;
        preference = 3;
        break;
    case '-':
        data = temp;
        type = 0;
        preference = 2;
        break;
    case '+':
        data = temp;
        type = 0;
        preference = 2;
        break;
    case '#':
        data = temp;
        type = 0;
        preference = 1;
        break;
    case ')':
        data = temp;
        type = 0;
        preference = 0;
        break;
    case '(':
        data = temp;
        type = 0;
        preference = 0;
        break;
    default:
        data = temp;
        type = 1;
        preference = 0;
        break;
    }
}

//Global Data.
stack <token> ops;
queue <token> nums;
token equation[50];

//Function prototypes.
void infixPostfix();
void evalPostfix();
token tokenMath(token, token, token);

//Main function to execute.
int main()
{
    string temp;
    string append = "!";

    //Asking for infix expression.
    cout << "Please input a infix equation:" << endl;
    getline(cin, temp);
    temp.append(append);

    //Tokenizing infix expression.
    for (int count = 0; count < temp.size(); count++)
    {
        equation[count].setData(temp[count]);
    }

    //Converting to postfix.
    infixPostfix();

    //Printing postfix version of equation.
    queue <token> print = nums;
    cout << "\nPostfix equation:" << endl;
    while (!print.empty())
    {
        print.front().outputData();
        print.pop();
    }
    cout << endl;

    //Finding the answer to the postfix expression.
    evalPostfix();

    //Printing equation answer.
    cout << "\nAnswer = " << ops.top().data - '0' << endl;
}

void infixPostfix()
{
    //Pushing dummy operator.
    token temp;
    temp.setData('#');
    ops.push(temp);

    int count = 0;

    //While not at end of expression.
    while (equation[count].data != '!')
    {
        if (equation[count].type == 1)
        {
               nums.push(equation[count]);
        }
        else if (equation[count].data == '(')
        {
               ops.push(equation[count]);
        }
        else if (equation[count].data == ')')
        {
               temp = ops.top();
               ops.pop();

               while (temp.data != '(')
               {
                      nums.push(temp);
                      temp = ops.top();
                      ops.pop();
               }
        }
        else
        {
               while (equation[count].preference <= ops.top().preference)
               {
                      temp = ops.top();
                      ops.pop();
                      nums.push(temp);
               }
               ops.push(equation[count]);
        }
        count++;
    }

    //Clean out the stack.
    while (ops.top().data != '#')
    {
        temp = ops.top();
        ops.pop();
        nums.push(temp);
    }
    ops.pop();
}

void evalPostfix()
{
    token temp;
    token num1;
    token num2;
    token op;
    token value;

    while (!nums.empty())
    {
        if (nums.front().type == 1)
        {
               temp = nums.front();
               nums.pop();
               ops.push(temp);
```

```
            }
            else
            {
                   num1 = ops.top();
                   ops.pop();
                   num2 = ops.top();
                   ops.pop();
                   op = nums.front();
                   nums.pop();
                   value = tokenMath(num1, num2, op);
                   ops.push(value);

                   //cout << ops.top().data << endl; //Debug line.
            }
        }
}
token tokenMath(token num1, token num2, token op)
{
    int iNum1 = num1.data - '0';
    int iNum2 = num2.data - '0';

    //cout << iNum1 << " " << iNum2 << endl; //Debug line.

    int result;
    token value;

    switch (op.data)
    {
    case'*':
        result = iNum1 * iNum2;
        value.setData(result + '0');
        return value;
        break;
    case'+':
        result = iNum1 + iNum2;
        value.setData(result + '0');
        return value;
        break;
    case'-':
        result = iNum1 - iNum2;
        value.setData(result + '0');
        return value;
        break;
    case'/':
        result = iNum2 / iNum1;
        value.setData(result + '0');
        return value;
        break;
    }
}
```

Output:

Using (1*2)+3 as input.

(1 of 4)

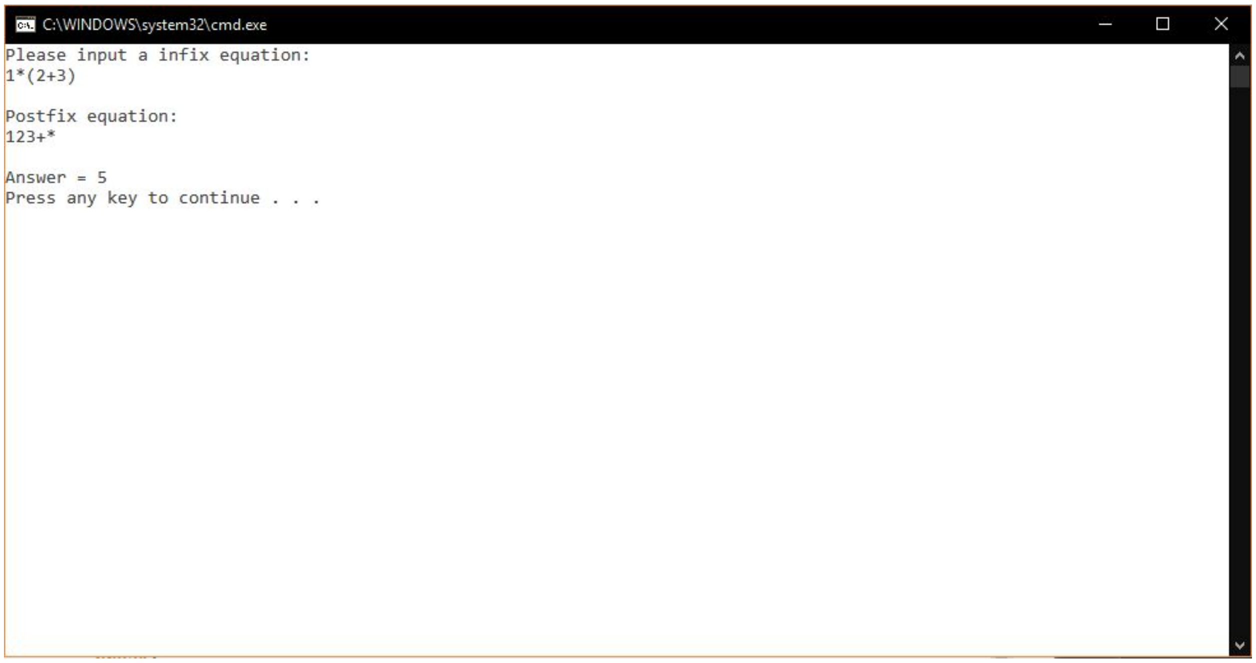


```
C:\WINDOWS\system32\cmd.exe
Please input a infix equation:
(1*2)+3

Postfix equation:
12*3+

Answer = 5
Press any key to continue . . .
```

Output:

Using 1*(2+3) as input.

(2 of 4)

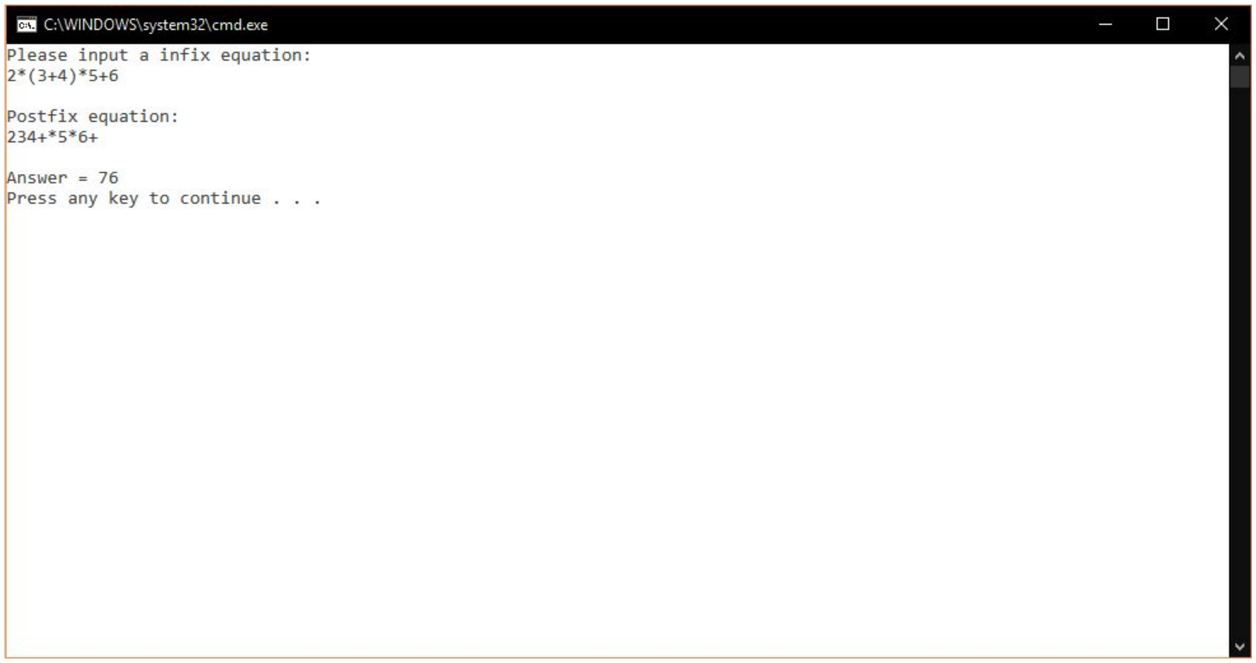


```
C:\WINDOWS\system32\cmd.exe
Please input a infix equation:
1*(2+3)

Postfix equation:
123+*

Answer = 5
Press any key to continue . . .
```

Output:

Using 2*(3+4)*5+6 as input.

(3 of 4)

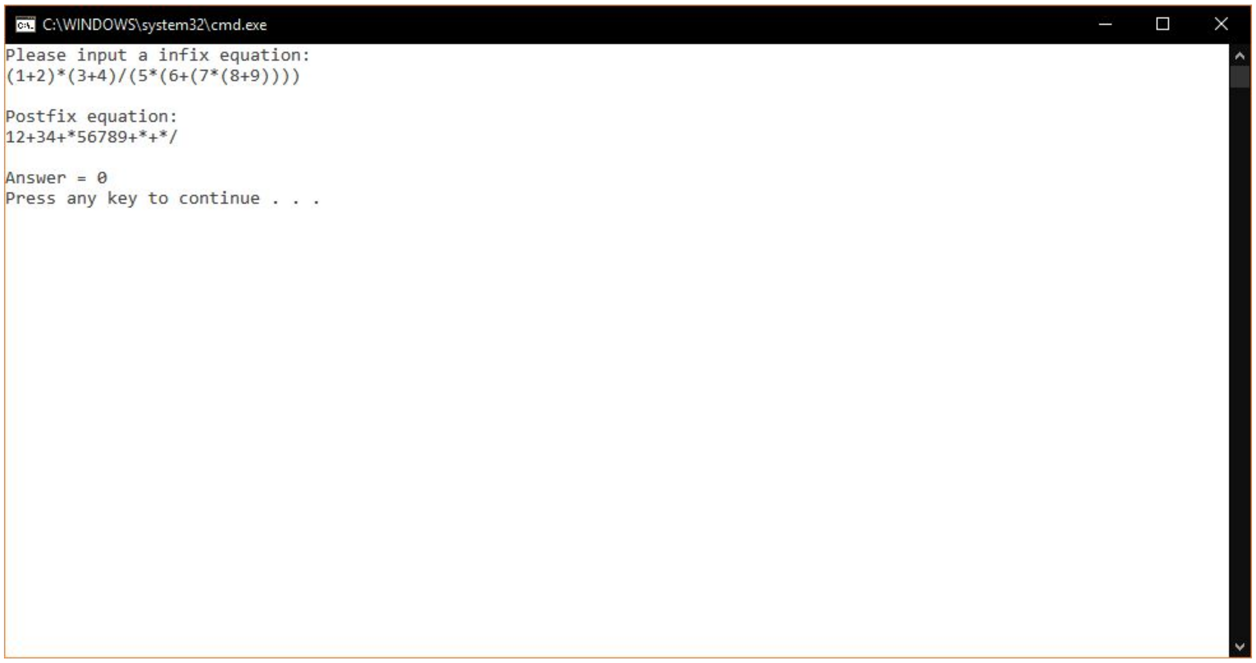


```
Please input a infix equation:
2*(3+4)*5+6

Postfix equation:
234+*5*6+

Answer = 76
Press any key to continue . . .
```

Output:

Using (1+2)*(3+4)/(5*(6+(7*(8+9)))) as input.

(4 of 4)



```
Please input a infix equation:
(1+2)*(3+4)/(5*(6+(7*(8+9))))

Postfix equation:
12+34+*56789+*+*/

Answer = 0
Press any key to continue . . .
```

Conclusion

I enjoyed this assignment. Actually implementing the code that I had been executing by hand was a relief. To see the computer do instantly what took me a substantial amount of time was also a great relief. Writing my own code for this assignment proved to be a little challenging, I kept making simple mistakes that prevented it from working correctly. However, as you can see, I got it working in the end.