CSC 2101 - Tutorial 3
Stack & Queue - Revisited

Due on October 12, 2011

Normaziah A. Aziz

Tutor - Iqram Mahmud

Contents

1 Converting Infix Expression to Postfix Using Stack

1.1 Procedure

Given an infix expression (i.e. `2*3-4/5`) we want to convert it into postfix (i.e. `23*45/-`).

- Scan the Infix string from left to right.
- Initialise an empty stack.
- If the scanned character is an operand, add it to the Postfix string. If the scanned character is an operator and if the stack is empty Push the character to stack.
  - If the scanned character is an operand and the stack is not empty, compare the precedence of the character with the element on top of the stack (stackTop). If stackTop has higher precedence over the scanned character Pop the stack else Push the scanned character to stack. Repeat this step as long as stack is not empty and stackTop has precedence over the character. Repeat this step till all the characters are scanned.
- (After all characters are scanned, we have to add any character that the stack may have to the Postfix string.) If stack is not empty add stackTop to Postfix string and Pop the stack. Repeat this step as long as stack is not empty.
- Return the Postfix string.

1.2 Implementation

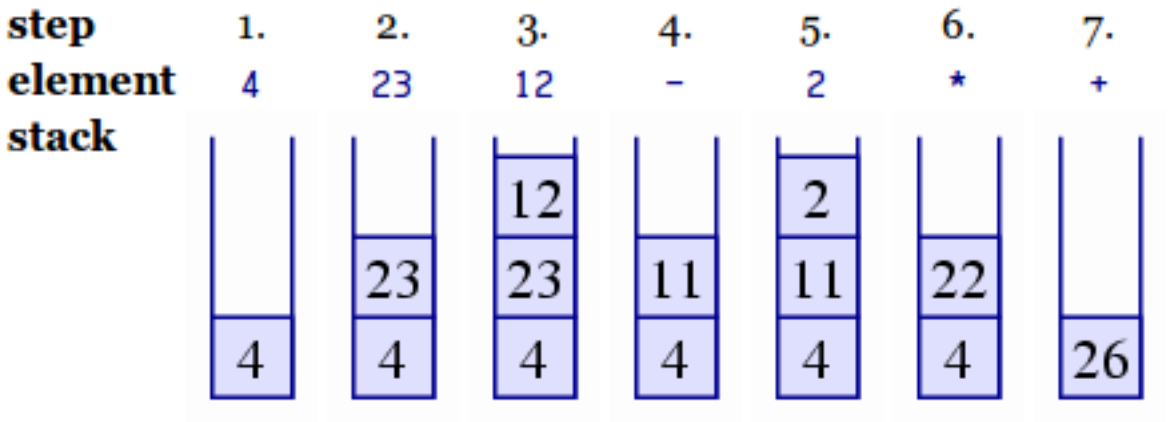
```
#include <iostream>
#include <cassert>
using namespace std;

class Stack {
    char a[100000];
    int limit;
    int top;

public:
    Stack() {
        top = 0;
        limit = 100000;
    }

    char pop() {
        assert( top > 0 );
        --top;
        return a[top];
    }

    char stackTop() {
```

```
        return a[top-1];
    }

    void push(char value) {
        a[top] = value;
        top ++;
    }

    bool isFull() {
        return top == limit;
    }

    bool isEmpty() {
        return top == 0;
    }
};

bool isoperator( char ch ) {
    if( ch == '+' || ch == '-' || ch == '/' || ch == '*' ) return true;
    else return false;
}

int priority( char ch ) {
    if( ch == '+' || ch == '-' ) return 1;
    if( ch == '*' || ch == '/' ) return 2;
    return -1;
}

string postfix( string in ) {
    Stack s;
    string ans = "";

    for(int i=0; i<in.size(); i++) {
        if( isoperator( in[i] ) ) {

            if( s.isEmpty() ) s.push( in[i] );
            else {
                while( !s.isEmpty() &&
                        priority( s.stackTop() ) > priority( in[i] ) )
                    ans += s.pop();
                s.push( in[i] );
            }
        }else ans += in[i];
    }

    while( !s.isEmpty() ) ans += s.pop();
    return ans;
}
```

```
int main() {
    string infix;
    cin >> infix;
    cout << postfix( infix ) << endl;
    return 0;
}
```

2 Evaluating a Postfix Expression Using Stack

This is how following postfix expression is evaluated -
4 23 12 - 2 * +



3 Stack

3.1 Problem A

Write a program that reads in a sequence of characters (that means one word) and prints them in reverse order. Use a stack.

3.2 Problem B

Write a program that reads in a sequence of characters, and determines whether its parentheses "balanced." (i.e. `(())` and `(()()())()` are balanced, while `))`, `()()(` and `(()()` are not)

Hint: for left delimiters, push onto stack; for right delimiters, pop from stack and check whether popped element matches right delimiter.

4 Queue

4.1 Problem C

A letter means enqueue and an asterisk means dequeue in the sequence
E A S * Y * Q U E * * * S T * * * I O * N * * *
Write down the sequence of values returned by the dequeue operations when this sequence of operations is performed on an initially empty FIFO queue.

For this problem it's not necessary to include the code.

5 Codes

Infix to postfix conversion	`http://codepad.org/wt5qAgFV`