MIT World Peace University

Fundamental Data Structures
Second Year B. Tech, Semester 1

Expression Evaluation using stack

Practical Report
Assignment 8

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 28, 2022

Contents

1 Aim

Writing a C Program to Evaluate a Postfix Expression using Stack.

2 Objectives

1. To study Stack and its operations
2. To study the importance of expression evaluation

3 Problem Statements

Write a C Program to Evaluate a Postfix Expression using Stack.

4 Theory

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : gcc on linux for C

6 Input

The Postfix Condition

7 Output

The Evaluation of the Postfix Expression entered.

8 Test Conditions

1. The Postfix Expression and Checking its Evaluation.

9 Code

9.1 Pseudo Code

9.2 C Implementation of Problem Statement

```c
#include<stdio.h>

#define MAX_SIZE 30
int stack[MAX_SIZE];
int top = -1;

int powerr(int a, int power){
    int i, result = 1;
    for(i = 0; i < power; i++){
        result *= a;
    }
    return result;
}

int isFull()
{
    if (top == MAX_SIZE - 1)
        return 1;
    else
        return 0;
}
int isEmpty()
{
    if (top == -1)
        return 1;
    else
        return 0;
}

int push(int item)
{
    if (!isFull())
    {
        top++;
        stack[top] = item;
    }
    else
    {
        printf("\nSTACK OVERFLOW!\n");
    }
}
int pop()
{
    if (isEmpty())
    {
        printf("Stack is Empty \n\n STACK UNDERFLOW!!");
        return 0;
    }
    else
    {
        // printf("Removed this thing %c\n", stack[top]);
        top--;
        return stack[top + 1];
    }
}


int calc(int op1, int op2, char op)
{
    int ans;
    switch (op)
    {
    case '+':
        ans = op1 + op2;
        break;
    case ’-’:
        ans = op1 - op2;
        break;
    case ’*’:
        ans = op1 * op2;
        break;
    case ’/’:
        ans = op1 / op2;
        break;
    case ’%’:
        ans = op1 % op2;
        break;
    case ’^’:
        ans = powerr(op1, op2);
        break;
    default:
        printf("\nInvalid operator");
        break;
    }
    return ans;
}

void eval(char post[MAX_SIZE])
{
    int z = 0, ans = 0, op1, op2;
    for (int i = 0; post[i] != 0; i++)
    {
        if(post[i] >= 97 && post[i] <= 122)
        {
            printf("\nEnter value of %c: ", post[i]);
            scanf("%d", &z);
            push(z);
        }
        else
        {
            op1 = pop();
            op2 = pop();
            ans = calc(op2, op1, post[i]);
            push(ans);
        }
    }
    printf("\nAnswer = %d\n", stack[top]);
}

int main()
{
    char post[30];
    printf("Enter the postfix expression: ");
    scanf("%s", post);
    eval(post);
}
```

Listing 1: Main.Cpp

9.3 Input and Output

```
Enter the postfix expression: abc+-cdb/+*b^c+

Enter value of a: 6

Enter value of b: 2

Enter value of c: 3

Invalid operator
Enter value of c: 3

Enter value of d: 8

Enter value of b: 2

Enter value of b: 2

Enter value of c: 3

Answer = 52
```

Listing 2: Output

10 Time Complexity

- **Insertion**: O(1)
- **Deletion**: O(1)
- **Searching**: O(n)

11 Conclusion

Thus, implemented postfix expression evaluation using stack data structure.

12 FAQs

1. **How prefix expression evaluation works?**

2. **What is the advantage of prefix and postfix expressions?**