Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mini Project Report

on ‘Postfix Evaluator’

[Code No: 202]

Submitted by:

Chandan Kumar Mahato (31)

Submitted to:

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

Submission Date: August 8, 2021, Sunday

Chandan Kumar Mahato

(31)

COMP 202: Data Structure and Algorithms
Mini Project

TASK

Write a program to evaluate postfix expression. What is the time complexity of your [redacted] (program)?

Evaluation of Postfix Expression.

Introduction

The postfix notation is used to represent algebric expressions. The expression written in posfix form are evaluated faster compared to infix notation as parenthesis are not required in postfix.

Algorithm

~~STEPS~~

Input: Postfix expression

Output: Result of Evaluation

STEPS

Step 1: Create and use a stack to store operands (or values).

Step 2: Scan the given expression and do following for every scanned element.

Step 2.1: If the element is number, push it into the stack

Step 2.2: If the element is a operator; pop two operands for the operator from stack. Evaluate the operator and push the result back to stack.

Algorithm contd...

Step 3: When the expression is ended, the number in the stack is the final answer.

Example:-

Evaluation of postfix expression:
"200 300 - 5 * 4 + 100 * 5 / 10 + 6 % 5 *"
is shown below with detailed explanation:-

Time Complexity of Algorithm:

Time complexity of evaluation is O(n) where n is the number of characters in input expression.
As we know, following algorithm evaluate the given expression scanning each character one by one and performing operation on them. Therefore, time complexity of evaluation is O(n)
n → number of characters in input expression.

Postfix Evaluator

Enter Valid Postfix Expression

Each Operand and Operator must be seprated by Space

200 300 - 5 * 4 + 100 * 5 / 10 + 6 % 5 *

Evaluate View Steps

Result = 10

Mini Project:- Postfix Evaluator

Name: Chandan Kumar Mahato

Group: Computer Engineering

Kathmandu University

Postfix Evaluator

Enter Valid Postfix Expression

Each Operand and Operator must be seprated by Space

200 300 - 5 * 4 + 100 * 5 / 10 + 6 % 5 *

Evaluate Reset

Result = 10

Steps of Postfix Evalution is Shown Below!

---

200 300 - 5 * 4 + 100 * 5 / 10 + 6 % 5 *

The character scanned is (**200**), which is an operand, so push it to the stack.

200

Stack

Expression...

---

200 **300** - 5 * 4 + 100 * 5 / 10 + 6 % 5 *

The character scanned is (**300**), which is an operand, so push it to the stack.

300

200

Stack

Expression...

---

200 300 **-** 5 * 4 + 100 * 5 / 10 + 6 % 5 *

The character scanned is (**-**), which is an operator, so pop its two operands from the stack.

Pop (**300**) from the stack for the right operand and then pop (**200**) from the stack to make the left operand

Empty

Stack

200-300=-100

Expression...

Next, push the result (**200-300=-100**) to the stack.

-100

Stack

Expression...

---

200 300 - **5** * 4 + 100 * 5 / 10 + 6 % 5 *

The character scanned is (**5**), which is an operand, so push it to the stack.

5

-100

Stack

Expression...

---

200 300 - 5 ***** 4 + 100 * 5 / 10 + 6 % 5 *

The character scanned is (*****), which is an operator, so pop its two operands from the stack.

Pop (**5**) from the stack for the right operand and then pop (**-100**) from the stack to make the left operand

Empty

Stack

-100*5=-500

Expression...

Next, push the result (**-100*5=-500**) to the stack.

-500

Stack

Expression...

---

200 300 - 5 * **4** + 100 * 5 / 10 + 6 % 5 *

The character scanned is (**4**), which is an operand, so push it to the stack.

4

-500

Stack

Expression...

---

200 300 - 5 * 4 **+** 100 * 5 / 10 + 6 % 5 *

The character scanned is (**+**), which is an operator, so pop its two operands from the stack.

Pop (**4**) from the stack for the right operand and then pop (**-500**) from the stack to make the left operand

Empty

Stack

-500+4=-496

Expression...

Next, push the result (**-500+4=-496**) to the stack.

-496

Stack

Expression...

---

200 300 - 5 * 4 + **100** * 5 / 10 + 6 % 5 *

The character scanned is (**100**), which is an operand, so push it to the stack.

100

-496

Stack

Expression...

---

200 300 - 5 * 4 + 100 ***** 5 / 10 + 6 % 5 *

The character scanned is (*****), which is an operator, so pop its two operands from the stack.

Pop (**100**) from the stack for the right operand and then pop (**-496**) from the stack to make the left operand

Empty

Stack

-496*100=-49600

Expression...

Next, push the result (**-496*100=-49600**) to the stack.

-49600

Stack

Expression...

---

200 300 - 5 * 4 + 100 * **5** / 10 + 6 % 5 *

The character scanned is (**5**), which is an operand, so push it to the stack.

5

-49600

Stack

Expression...

---

200 300 - 5 * 4 + 100 * 5 **/** 10 + 6 % 5 *

The character scanned is (**/**), which is an operator, so pop its two operands from the stack.

Pop (**5**) from the stack for the right operand and then pop (**-49600**) from the stack to make the left operand

Empty

Stack

-49600/5=-9920

Expression...

Next, push the result (**-49600/5=-9920**) to the stack.

-9920

Stack

Expression...

---

200 300 - 5 * 4 + 100 * 5 / **10** + 6 % 5 *

The character scanned is (**10**), which is an operand, so push it to the stack.

10

-9920

Stack

Expression...

---

200 300 - 5 * 4 + 100 * 5 / 10 **+** 6 % 5 *

The character scanned is (**+**), which is an operator, so pop its two operands from the stack.

Pop (**10**) from the stack for the right operand and then pop (**-9920**) from the stack to make the left operand

Empty

Stack

-9920+10=-9910

Expression...

Next, push the result (**-9920+10=-9910**) to the stack.

-9910

Stack

Expression...

---

200 300 - 5 * 4 + 100 * 5 / 10 + **6** % 5 *

The character scanned is (**6**), which is an operand, so push it to the stack.

6

-9910

Stack

Expression...

---

200 300 - 5 * 4 + 100 * 5 / 10 + 6 **%** 5 *

The character scanned is (**%**), which is an operator, so pop its two operands from the stack.

Pop (**6**) from the stack for the right operand and then pop (**-9910**) from the stack to make the left operand

Empty

Stack

-9910%6=2

Expression...

Next, push the result (**-9910%6=2**) to the stack.

2

Stack

Expression...

---

200 300 - 5 * 4 + 100 * 5 / 10 + 6 % **5** *

The character scanned is (**5**), which is an operand, so push it to the stack.

5

2

Stack

Expression...

---

200 300 - 5 * 4 + 100 * 5 / 10 + 6 % 5 *****

The character scanned is (*****), which is an operator, so pop its two operands from the stack.

Pop (**5**) from the stack for the right operand and then pop (**2**) from the stack to make the left operand

Empty

Stack

2*5=10

Expression...

Next, push the result (**2*5=10**) to the stack.

10

Stack

Expression...

Hence, After Scanning all the characters in the give postfix expression. The final element in the stack is our result.