Page No.
Date

Devanshu Surana
1032210755, PC - 12
FDS Lab Assignment - 8

Problem Statement:
Write a C program to evaluate postfix expression using stack.

Objectives:
1. To study stack and its operation
2. To study importance of expression evaluation.

Theory:
An expression is evaluated based on the operator procedure and associativity. The operator with higher precedence is evaluated first and with least precedence at last.
Arithmetic expression can be written in one of the three forms
i) Infix expression
ii) Prefix expression
iii) Postfix expression
Infix expression are harder computers to evaluate because of additional work needed to decide precedence of while prefix and postfix expression are easy as no precedence or or order is required. To evaluate expression using stacks, we have to main two stacks - operator stack and operand stack. If character is an operand then we push it to operand stack and if character is an operator, we pop two operands from stack, operate on them and push the result onto stack.
For example -
Postfix expression → 748 * +

Character	Operation	stack	Calculation
7	push	7	
4	push	7, 4	
8	push	7, 4, 8	
*	pop (2 elements) and evaluate	7	4 * 8 = 32
+	push result 32	7, 32	
	pop (2 element)		7 + 32 = 39
	Evaluate push result 39	39	
	pop()		39 (Result)

Platform: 64 bit Open source Linux or its derivatives
- Open Source C programming tool like gcc/ Eclipse Editor)

PSEUDO Code: Write a pseudo code postfix exp^n evaluation.

```
void evaluate (E) {
int (stack) s;
for (i = 1 to n) {
  if (E [i] is an operator ) (for ex) '+')
      op1 = S.pop();
      op2 = S.pop;
     value = op1 + op2
      S.push (value);
  else
    S.push (E [i]);
  }
}
```

Page No.
Date

Time Complexity: O(n)

Conclusion

Thus, implemented postfix expression evaluation using stack data structure.

FAQ's.

Ans1] a) We have to maintain 2 stacks -(i) operand (ii) operator
b) Read first character of expression
c) if character is operand push it to operand stack else pop(2) elements from stack, evaluate them and store back result from stack.
d) Read step b & c until stack is empty or whole expression is evaluated

Ans2] Prefix expression and postfix expression are easy for computers to evaluate because no precedence of operator is required in this. This is the biggest advantage of prefix expression and postfix expression.

21/2/22