**Kathmandu University**

**Department of Computer Science and Engineering**

**Dhulikhel, Kavre**



**A Report on**

**COMP 202: Data Structures and Algorithms**

**Mini Project**

**Submitted by:**

Bibek Mishra (33)

Gautam Buddha Shakya (41)

**Submitted to:**

Dr. Rajani Chulyadyo

Department of Computer Science and Engineering

**Submission Date:**

January 15, 2019

**Task:**  To write a program to evaluate postfix expression and to find the time complexity of the program

To evaluate postfix expression, we used the stack data structure.

Note: Here we considered only {+, −,∗,/} operators.

**Implementation:**

Algorithm

Input: A valid postfix expression

Output: An equivalent solution

Steps:  
1. Initialize a stack.  
2. For each character input in expression  
3. If input is an operand   
…..3.1 push input into stack

4. Else if,input is operator

5. a=pop(stack)  
6. b=pop(stack)  
7. Evaluate a and b with operator as c=b<input>a and push c to stack.  
8. end if   
9. end for  
10. Return final stack element=pop(stack)

**C++ Source Codes**

The C++ source codes are submitted with the labels stack.h, stack.cpp and main.cpp.

The stack.h and stack.cpp handle the stack operations such as push, pop, peak, etc. The main.cpp includes the stack implementation and takes postfix expression input from the user to solve it in the same manner as the algorithm above.

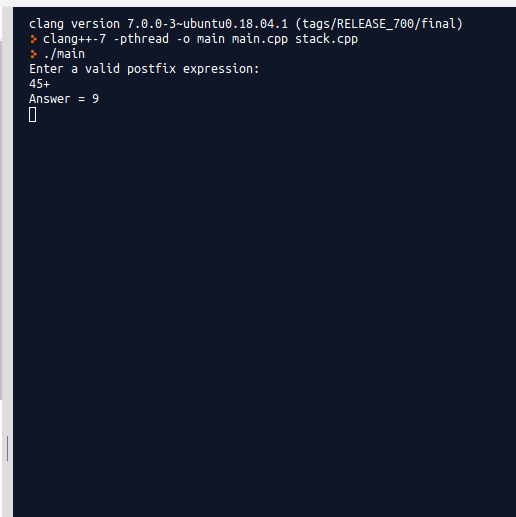
**Test Program** (Manually done according to the algorithm)

**Input**: 54+63-/ : an infix expression

|  |  |
| --- | --- |
| **Token** | **Stack** |
| 5 | 5 |
| 4 | 4 5 |
| + | 5+4=9 |
| 6 | 6 9 |
| 3 | 3 6 9 |
| - | 6-3=3 9 |
| / | 9/3=3 |
|  | Result=3 |

**Expected Output**: 54+63-/ : 3

**Output from the program:**



**\**

Hence, the implementation works correctly.

**Time Complexity of the Program**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Program** | **Steps** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | int main()  {  int i=0,x,a,b,c;  char input[100];  Stack s1;  cout<<"Enter a valid postfix expression:"<<endl;  cin>>input;  while(input[i]!='\0')  {  if(input[i]>='0' && input[i]<='9')  {  x=input[i]-48;  s1.push(x);  }  else if (input[i]=='+' || input[i]=='-' || input[i]=='\*' || input[i]=='/')  {  a=s1.pop();  b=s1.pop();  switch(input[i])  {  case '+':  c=b+a;  s1.push(c);  break;  case '-':  c=b-a;  s1.push(c);  break;  case '\*':  c=b\*a;  s1.push(c);  break;  case '/':  c=b/a;  s1.push(c);  break;  }  }  i++;  }  cout<<"Answer = "<<s1.pop()<<endl;  return 0;  } | 0  0  1  0  0  1  1  N+1  0  N  0  C1  C1  0  C2  0  C2  C2  C2  0  C3  C3  C3  C3  C4  C4  C4  C4  C5  C5  C5  C5  C6  C6  C6  C6  0  0  N  0  1  1  0 |
|  | TOTAL | N+K |

Hence, Time complexity or the total runtime of the program = O(N).

Here,

C1= No. of loops to check operand

C2= No. of loops to check operator

C3= No. of loops to check ‘+’ operator

C4= No. of loops to check ‘-’ operator

C5= No. of loops to check ‘\*’ operator

C6= No. of loops to check ‘/’ operator

Simply, if we assume the length of the postfix expression input from the user to be ‘N’, then there will be a single loop in the program accompanied by several other if, else and switch statements. So,when we have a loop for N times, the time complexity of the program would be O(N).