**Experiment No.2a**

**Title:** Evaluation of Postfix Expression using C

**Problem Statement:**

Program to Evaluate a Postfix Expression by obtaining the result of the expression.

**Algorithm:**

**Step 1**: Create a stack to store operands.

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

1. If the element is a number, push it into the stack.
2. If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack

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

**Code:**

#include<stdio.h>

#include<ctype.h>

int top=-1;

int stack[20];

void push(int x)

{

stack[++top]=x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int op1,op2,num,res;

printf("Enter the expression:");

scanf("%s",exp);

e=exp;

while(\*e != '\0')

{

if (isdigit (\*e))

{

num=\*e-48;

push(num);

}

else

{

op1=pop();

op2=pop();

switch(\*e)

{

case '+':res=op1+op2;

break;

case '-':res=op2-op1;

break;

case '\*':res=op1\*op2;

break;

case '/':res=op2/op1;

break;

}

push(res);

}

e++;

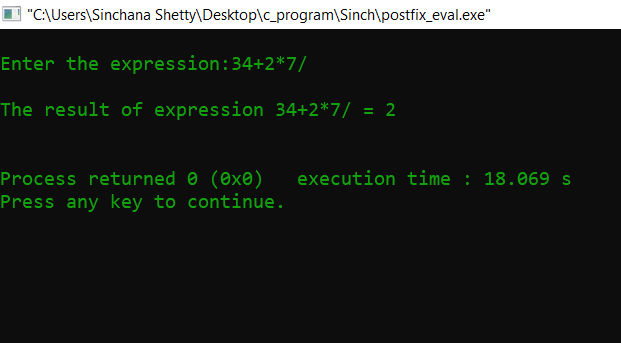
}

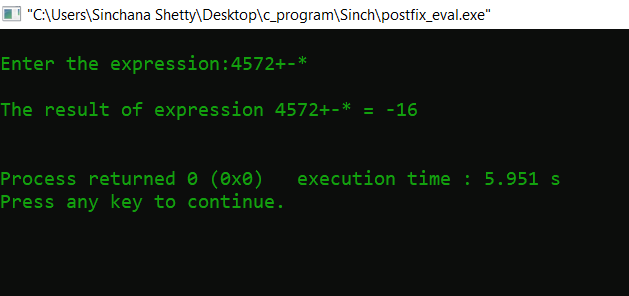
printf("The result of expression %s=%d\n\n",exp,pop());

return 0;

}

**Results:**





**Analysis(Limitations):**

Usage of stack to evaluate postfix expression is easier as we have to scan the expression from left to right only once. But it has some limitations and we have to assume the following:

* Let us assume that each line is in the form of a string of digits and operator symbols and that the operands are single nonnegative digits.
* Another assumption is that the expression ends with null character (‘\0’) as we read it as a string.