

**Experiment no 3:Evaluation of postfix Expression using stack ADT**

**NAME: SHREYA SANJAY CHAUDHARI**

**BRANCH: COMPS DIV: 3 ROLL NO: 04**

**Aim:** Implementation of Evaluation of Postfix Expression using stack ADT

**Objective:**

* Understand the use of stack
* Understand importing an ADT in an application program
* Understand the instantiation of stack ADT in an application Program
* Understand how the member function of an ADT are accessed in an application program

**Theory:**

Postfix evaluation is a stack-based method to compute the result of a mathematical expression in postfix notation. Here's the theory in its shortest form:

1. \*\*Stack Setup\*\*: Start with an empty stack.

2. \*\*Scan Expression\*\*: Read the expression from left to right.

3. \*\*Operand Handling\*\*: If operand, push onto stack.

4. \*\*Operator Handling\*\*: If operator, pop operands (usually 2), perform operation, push result.

5. \*\*Final Result\*\*: Result is the only item left on the stack.

This concise outline captures the essence of postfix evaluation. It simplifies the process to its core steps: pushing operands, applying operators, and obtaining the final result using a stack.

**Algorithm:**

1.#

2. item = READ\_Symbol()

3. while item != ‘#’ do

if item=operand then

push(item)

else

operand = item

y = pop()

x = pop()

t = xpopy

PUSH(t)

end if

item = READ Symbol()

end while

4. value = pop()

5. stop

**Code :**

#include<stdio.h>

int stack[20];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

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

{

if(isdigit(\*e))

{

num = \*e - 48;

push(num);

}

else

{

n1 = pop();

n2 = pop();

switch(\*e)

{

case '+':

{

n3 = n1 + n2;

break;

}

case '-':

{

n3 = n2 - n1;

break;

}

case '\*':

{

n3 = n1 \* n2;

break;

}

case '/':

{

n3 = n2 / n1;

break;

}

}

push(n3);

}

e++;

}

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

return 0;

}

**output:**



**Conclusion :**

In summary, postfix evaluation offers an elegant solution to evaluate mathematical expressions efficiently and accurately, making it a valuable tool for computations across different domains.