

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

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

**Objective:**

1. Understand the use of stack
2. Understand importing an ADT in an application program
3. Understand the instantiation of stack ADT in an application Program
4. Understand how the member function of an ADT are accessed in an application program

**Theory:**Postfix Expression evaluation using a stack ADT involves utilizing a stack data structure to process and calculate the value of the expression.The stack helps to manage the order of operation and operands,ensuring correct evaluation.This notation style is known as Reversed Polish Notation.In this notation style, the operator is postfixed to the operands i.e., the operator is written after the operands.

For example, ab+.

This is equivalent to its infix notation a + b.

**Algorithm:**

1) Add ) to postfix expression.

2) Read postfix expression Left to Right until ) encountered

3) If operand is encountered, push it onto Stack

[End If]

4) If operator is encountered, Pop two elements

i) A -> Top element

ii) B-> Next to Top element

iii) Evaluate B operator A

push B operator A onto Stack

5) Set result = pop

6) END

**Code :**

#include<stdio.h>

#include<conio.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++;

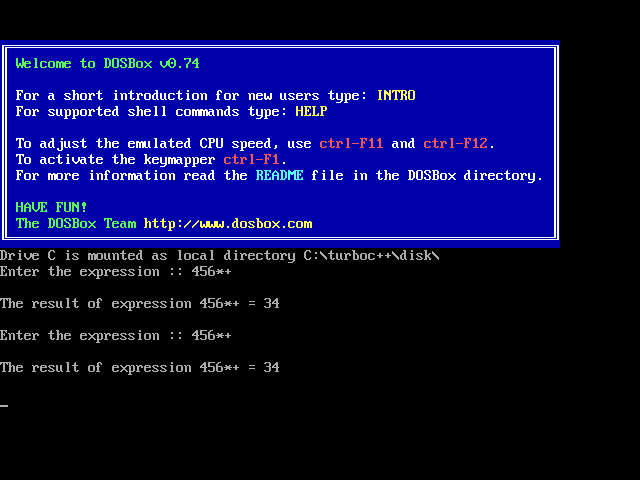
}

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

getch();   
return 0;

}

**Output:**



**Conclusion :**

The process involves systematically scanning expression, pushing operands onto the stack and performing calculation with operations.The stack’s Last-In-First-Out behavior ensures that the correct order of operations is maintained, even in complex expressions.